EFFICIENT AND PRINCIPLED ROBOT LEARNING: THEORY AND ALGORITHMS

A Dissertation
Presented to
The Academic Faculty

By

Ching-An Cheng

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Robotics

School of Interactive Computing
Georgia Institute of Technology

May 2020

EFFICIENT AND PRINCIPLED ROBOT LEARNING:
THEORY AND ALGORITHMS

Approved by:

Dr. Byron Boots, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Dr. Evangelos A. Theodorou
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Geoff Gordon
Machine Learning Department
Carnegie Mellon University

Dr. Karen Liu
School of Engineering
Stanford University

Date Approved: December 5, 2019

The final test of a theory is its capacity to solve the problems which originated it.

George Dantzig

To my wife, my parents, and my cats.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Byron Boots for his continuous support in both research and life. Especially, I truly appreciate the research freedom and invaluable advices he provided throughout my Ph.D. study, which have cultivated me to become an independent researcher.

Besides my advisor, I would like to thank my thesis committee: Prof. Seth Hutchinson, Prof. Evangelos A. Theodorou, Prof. Geoff Gordon, and Prof. Karen Liu for their encouragement and insightful comments that further widen my research in various perspectives. I would like to equally thank my mentors, Dr. Nathan Ratliff, Prof. Dieter Fox, Dr. Alekh Agarwal, and Dr. Andrey Kolobov, who I was privileged to work with during internships and have provided me priceless suggestions ever since.

Tremendous thanks also go to my wonderful friends, colleagues, and collaborators; especially, Xinyan Yan, Amirreza Shaban, Nolan Wagener, Anqi Li, Jacob Sacks, Jonathan N. Lee, Dr. Mustafa Mukadam, Dr. Yungpeng Pan, Dr. Hugh Salimbeni, Dr. Remi Tachet des Combes, Dr. Stan Birchfield, Prof. Marc Deisenroth, Prof. Magnus Egerstedt, and Prof. Ken Goldberg. I learned a significant amount from these collaborations and fruitful discussions. This thesis would not be possible without their contributions.

Last but not the least, I would like to thank my family, especially my wife Jiashan Chen and my cats (Alice and Ramesses). They have been my mental support, providing me unconditioned love throughout good and bad times of this journey. I would not be where I am today if not for them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Roboticists have long envisioned fully-automated robots that can operate reliably in unstructured environments. This is an exciting but extremely difficult problem; in order to succeed, robots must reason about sequential decisions and their consequences in face of uncertainty. As a result, in practice, the engineering effort required to build reliable robotic systems is both demanding and expensive. This research aims to provide a set of techniques for efficient and principled robot learning. We approach this challenge from a theoretical perspective that more closely integrates analysis and practical needs. These theoretical principles are applied to design better algorithms in two important aspects of robot learning: policy optimization and development of structural policies. This research uses and extends online learning, optimization, and control theory, and is demonstrated in applications including reinforcement learning, imitation learning, and structural policy fusion. A shared feature across this research is the reciprocal interaction between the development of practical algorithms and the advancement of abstract analyses. Real-world challenges force the rethinking of proper theoretical formulations, which in turn lead to refined analyses and new algorithms that can rigorously leverage these insights to achieve better performance.

CHAPTER 1
INTRODUCTION

We have long envisioned a world with fully-automated agents that can operate reliably to solve complex tasks in unstructured, real-world environments. This is an exciting but extremely difficult problem: in order to succeed, an agent must reason about sequential decisions and their consequences through a careful balance between exploration and exploitation. The agent should adapt to new environments to maximize performance, while also making decisions that contend with the agent's uncertainty about its own experiences of the world. Broadly speaking, this problem can be framed as reinforcement learning (RL), where the interactions between the agent and the environment follow the rules of some (contextual/partially observable) Markov decision process. However, despite our decades-long research on RL and understanding of near-optimal algorithms, there still exists a significant gap between theory and practice.

1.1 The Reality Gap

This "reality gap" can be attributed, at least in part, to the mismatch between the formal problem settings commonly studied by theoreticians and the domain properties of real-world applications. As a result, engineers often need to heuristically modify existing theoretical frameworks via trial and error to make them work well in practice, which can be a time-consuming and expensive process.

Nevertheless, the effectiveness of modifying these frameworks by hand is ultimately compromised by the lack of integration across various methodologies and theories that have been implicitly adopted throughout the design pipeline. Because theoretical tools are often designed without the practical applications in mind, they can be overly pessimistic or optimistic when deployed on real systems. On the one hand, when an overly simplified

problem setup is taken, such as breaking the full policy optimization problem into isolated parts (like planning, perception, control, etc.), the agent can suffer from uncontrolled performance bias due to optimistic assumptions. On the other hand, when we approach real-world applications using a overly general problem setup, such as black-box learning, the agent can also have suboptimal performance. While this approach does not make optimistic assumptions, they can be too pessimistic about the problem's difficulty. Therefore, algorithms designed based on such a general setting will waste efforts to tackle imaginary corner cases that do not exist in the source applications, making the agent learn less data efficiently.

Indeed, when an algorithm with strong theoretical guarantees is applied to a real-world problem that well matches its abstract setup, high-performance systems can be built (such as contextual bandit and recommendation systems). But, often this is not the case. In practice, we have been unconsciously using *shallow* human learning to figure out the last mile of design, seldomly using the real-world insights to refine our theories *end-to-end*. Errors due to the mismatch between theory and practice therefore compound in each stage of design, leading to the reality gap.

When systems become more complicated, this reality gap can widen. As an example, most robots are built on a hierarchical decomposition principle, where complex tasks are decomposed into simpler problems. However, the untold story behind the success of hierarchical approaches is the ad-hoc way in which these simpler components are combined and tuned: a planning module may have been designed under perfect realization assumptions, and yet imperfections due to control, modeling errors, and computational budget unavoidably arise in a real system. Consequently, the engineering effort required to build robotic systems with desired performance is both demanding and expensive.

1.2 Theories Driven by Practical Needs

In this thesis, I propose to bridge the reality-gap from a theoretical perspective that more closely integrates analysis and practical needs: *Real-world challenges force the rethinking of proper theoretical formulations, which in turn lead to refined analyses and new algorithms that can rigorously leverage these insights to achieve better performance.* While theory cannot replace engineering and empirical studies, a proper theoretical setup can help provide a panoramic perspective on the underlying difficulties and indicate systematic directions for improvement. I will demonstrate the efficacy of this approach through my research to date, which contributes a set of techniques for efficient and principled robot learning. This thesis focuses on two important aspects of robot learning: policy optimization and development of structural policies, where a reciprocal interaction is shown between the development of practical algorithms and the advancement of abstract analyses.

1.3 Policy Optimization: An Online Learning Approach

Policy optimization concerns learning the decision rule (i.e. the policy) for sequential decision-making tasks based on interactions and experiences. My research takes an *online learning* approach to policy optimization, in which the policy update process is abstracted into the interactions between two players (a learner and an opponent) (Cheng and Boots, 2018; Cheng et al., 2018a, 2019a,b,c,d; Pan et al., 2019; Pan et al., 2018). The use of online learning to analyze policy optimization was pioneered by Ross, Gordon, and Bagnell (2011) who proposed to reduce imitation learning (IL) to adversarial online learning. The research here (constituting Part I and Part II of the thesis) builds on top of these ideas and provides further insights.

1.3.1 Efficient Policy Optimization via Online Imitation Learning

Policy optimization in practice is not simply about solving a generic RL problem as it is mathematically stated, because specific information is often known about the sequential decision problem of real-world applications. For example, a robot does not need to learn to move from scratch, as similar problems have been studied for decades in motion planning and optimal control. Therefore, we should also consider this prior knowledge as part of the overall problem description, and an important aspect of policy optimization is about designing proper learning paradigms through which a good policy can be learned efficiently (Chapter 2).

One instantiation of this idea is IL, which leverages domain knowledge in the form of expert demonstrations to build surrogate problems that have nicer optimization properties (Chapter 3). The expert in IL can be fairly general. While experts are commonly associated with human demonstrators, they can also be heuristics or suboptimal algorithms (like planning in a simplified world). That is, we should more appropriately view IL as a mechanism to transfer our prior knowledge about the problem (abstracted in the form of an expert policy) to quickly instantiate non-trivial policies.

We put this design principle into practice: In Chapter 4, we build an autonomous system that is capable of using cheap on-board sensors on a rally car to perform high-speed maneuvers on off-road terrain (Pan et al., 2019; Pan et al., 2018). This is a challenging policy optimization problem, where observations are high-dimensional and dynamics are highly stochastic, and running a usual RL algorithm here could risk permanently damaging the car. We accomplish this feat by instead leveraging IL to let the robot mimic a model predictive control (MPC) algorithm that runs with expensive compute and sensor resources, the prior knowledge about the problem. To this end, we take online IL (Ross, Gordon, and Bagnell, 2011) and batch IL (i.e. behavior cloning) (Pomerleau, 1989) as potential candidates. We show in theory that when the expert is stable and can provide consistent feedback to the learner for any query, online IL enjoys a better trade-off between the expert-learner

performance gap and sample efficiency. Importantly, this theoretical insight translates into our real-wold system: the policy trained with online IL learned to drive the rally car faster and generalized better to new conditions.

While the idea of expressing prior information as expert policies is powerful, IL can yield policies with inferior performance compared to RL when the expert policy is suboptimal. To combine the best aspects of RL and IL, in Chapter 5, we frame several popular RL and IL algorithms in a common mirror descent framework and propose a simple randomized strategy, called LOKI (Locally Optimal search after K-step Imitation) (Cheng et al., 2018a). LOKI first performs a small but random number of first-order online IL iterations before hard switching to a policy gradient RL method. Both theoretically and empirically, we show not only can LOKI improve faster than common RL methods, but it can also significantly outperform a suboptimal expert. The insight is that training a policy with online IL generally converges faster than with RL in the early stage, as it minimizes a relative performance measure and is less noisy.

1.3.2 Convergence and Acceleration of Online Imitation Learning

Can we further accelerate online IL? Inspired by this question, we revisit the reduction made by Ross, Gordon, and Bagnell, 2011 which defines the per-round loss in online learning as the expectation of some performance upper bound over the distribution of states visited by the current policy. This reduction treats the state distribution as an online, adversarial component, and all the algorithms discussed above and in the literature had been designed based on this principle. But is policy optimization truly adversarial? We found that in practice it is not: the policies are executed in the same environment whose dynamics do not change adversarially with policies; instead certain continuity properties exist. In other words, the per-round losses in online IL are actually predictable from the past information.

With this insight, in Chapter 6, we prove the first last-iterate convergence result for a

family of online IL algorithms, called value aggregation (Ross and Bagnell, 2014). While the existence of a good policy in the policy sequence can be guaranteed non-asymptotically using the traditional adversarial treatment, little was known about the convergence of the sequence or the performance of the last policy. We debunk the common belief that value aggregation always produces a convergent policy sequence with improving performance. Moreover, we identify a critical stability condition for convergence and provide a tight non-asymptotic bound on the performance of the last policy (Cheng and Boots, 2018). These new theoretical insights show that online IL can be stabilized by regularization techniques, including the heuristic use of mixture policies in the literature.

The above discovery leads to a new family of algorithms, called MoBIL (Model-Based Imitation Learning), that combines the knowledge about the predictable components inside online IL to achieve provably faster and unbiased convergence (Cheng et al., 2019b). In Chapter 7, we propose two model-based IL algorithms inspired by Follow-the-Leader (FTL) with prediction: MoBIL-VI based on solving variational inequalities (VIs) and MoBIL-Prox based on stochastic first-order updates. These two methods leverage a *predictive model* to estimate future gradients in lieu of real interactions with the environment to speed up policy learning. For example, a predictive model can be built using a biased simulator or past experiences. When the predictive model is learned online, these algorithms can provably accelerate the best known convergence rate up to an order.

1.3.3 Toward a Generic Framework for Policy Optimization and Beyond

We further extend the above ideas beyond IL to a much larger class of problems. We generalize these findings along two directions, proposing 1) a refined online learning setup, called Continuous Online Learning (COL), that intrinsically models the regularity of loss functions across rounds (Cheng et al., 2019c); and 2) a meta algorithm, called PICCOLO (PredICtor-COrrector onLine Optimization[1]), that leverages this regularity to turn a stan-

[1] In the original paper, it was named PredICtor-COrrector poLicy Optimization.

dard online learning algorithm designed for adversarial problems into a new hybrid algorithm with accelerated convergence (Cheng et al., 2019d).

Introduced in Chapter 8, COL aims to bridge the reality gap between the classic adversarial setup of online learning and the applications with such regularity properties. Toward this end, COL refines the relationship between regret, per-round losses, and feedback information. In COL, the regret is measured with respect to a loss sequence whose gradients change continuously across rounds with respect to the learner's decisions, while the learner receives potentially adversarial feedback about these regular online losses (such as stochastic gradients). COL appropriately describes many interesting applications, from general equilibrium problems (EPs) to optimization in episodic MDPs, where online losses are predictable (in expectation). For example in online IL, the regret is measured in terms of the generalization error of imitation, which is continuous because of the expectation over the state distribution induced by the learner's policy, whereas the feedback returned to the learner is only its noisy finite-sample approximation. Structural prediction, system identification, and fitted Q-learning can be framed similarly as well.

COL formalizes these regularity properties observed in practice into an abstract theoretical setup, so that we can better study how this extra information changes the learning characteristics. In Chapter 8, we prove a fundamental relationship between COL and EPs: On the one hand, all monotone EPs admit a reduction to achieving sublinear static regret in COL. On the other hand, achieving sublinear dynamic regret in COL is equivalent to solving all EPs. Using this insight, we offer conditions for efficient algorithms that achieve sublinear dynamic regret, even when the losses are chosen adaptively without any *a priori* variation budget. Furthermore, we show for COL a reduction from dynamic regret to both static regret and convergence in the associated EP, allowing us to analyze the dynamic regret of many existing algorithms.

As a compelling demonstration of the new COL framework, in Chapter 9, we present a reduction from RL to no-regret online learning based on the saddle-point formulation of

RL, by which *any* online algorithm with sublinear regret can generate policies with provable performance guarantees. This new perspective decouples the RL problem into two parts: regret minimization and function approximation. The first part admits a standard online-learning analysis, and the second part can be quantified independently of the learning algorithm. Therefore, the proposed reduction can be used as a tool to systematically design new RL algorithms. We demonstrate this idea by devising a simple RL algorithm based on mirror descent and the generative-model oracle. For any γ-discounted tabular RL problem, with probability at least $1-\delta$, it learns an ϵ-optimal policy using at most $\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}{(1-\gamma)^4\epsilon^2}\right)$ samples. Furthermore, this algorithm admits a direct extension to linearly parameterized function approximators for large-scale applications, which has computation and sample complexities independent of $|\mathcal{S}|$,$|\mathcal{A}|$, though at the cost of potential approximation bias.

Complementary to this refined COL setup, in Chapter 10, we provide a constructive framework, PICCOLO (Cheng et al., 2019d), to design algorithms that leverage the predictability in losses to speed up learning. PICCOLO is a meta algorithm. An instance algorithm returned by PICCOLO uses the concept of predictive models presented in Chapter 7 to optimize online decisions. It recursively repeats the two following steps: In the Prediction Step, the learner uses a predictive model to estimate the gradient of the next loss function and then applies the predictions, potentially through solving a VI or an optimization problem, to update the decision; in the Correction Step, the learner runs the updated decision in the environment, receives (noisy) gradient feedback, and then corrects the decision using the gradient error.

The design of PICCOLO was made possible by a novel reduction that converts a given predictable online learning problem into a new adversarial problem. We prove that PICCOLO can improve the static regret rate of any base algorithm that can be written as mirror descent or FTRL (Follow-the-Regularized-Leader).[2] An intuition for the faster learning

[2]MoBIL in Chapter 7 is a special case of PICCOLO when the base algorithm is FTRL.

rate is that PICCOLO decouples optimization complexity from sample complexity (measured each round) through embedding complex computation into the Prediction Step.

We can use the PICCOLO idea in policy optimization by identifying each online decision as a policy. Because the family of mirror descent and FTRL update rules is rich and covers most first-order algorithms used in RL and IL (cf. Chapter 5), we can use PICCOLO to speed up many policy optimization techniques. At a high level, we can treat PICCOLO as a hybrid algorithm that combines a model-based update (Prediction Step) and a model-free update (Correction Step), where the concept of predictive model provides an abstraction of model information (i.e. past experiences). We show that PICCOLO does not suffer from policy performance bias due to modeling error, unlike the classic model-based approaches. Therefore, PICCOLO provides a systematic framework for designing new algorithms that can safely leverage imperfect models. To corroborate the theory, in Chapter 10 we construct a predictable online learning setup for RL, and we 'PICCOLOed' multiple algorithms in simulation, including ADAM (Kingma and Ba, 2014), natural gradient descent (Kakade, 2002), and trust-region optimization (Schulman et al., 2015b). The experimental results show that the PICCOLOed versions consistently surpassed the base algorithm and were robust to predictive model errors; they converged even when the models were adversarial.

1.4 Expressive Structural Policies with Stability Guarantees

Another aspect of this thesis (Part III) focuses on designing the policy structure in policy optimization. While universal function approximators can be used to parameterize policies, policies learned in this way do not have robust guarantees about important qualities such as stability and interpretability. To reliably apply learning techniques to robotics, we must find a way to ensure these properties into our learned policies in a *non-statistical* manner.

We treat this requirement as a design problem of structuring a policy class in which every policy is interpretable and stable (Cheng et al., 2018b; Li et al., 2019b; Mukadam et al., 2019), so that statistical policy learning techniques (e.g. the algorithms described in

Parts I and II of the thesis) can be freely applied. Although parameterizing a stable class of policies is not difficult, an ideal policy class must also be flexible and expressive enough to achieve desired performance.

1.4.1 A Geometric Framework for Policy Fusion

We demonstrate this idea in the design of second-order policies, i.e. policies that control acceleration/torque based on feedback fo position and velocity. In Chapter 11, we propose a tree-structured computational graph, called RMPflow, that can efficiently and consistently combine multiple subtask policies into a global policy (Cheng et al., 2018b). Based on differential geometry, RMPflow uses the Riemannian Motion Policy (RMP) framework (Ratliff, Issac, and Kappler, 2018) to represent policies on manifolds and has a set of operators to propagate RMPs across different manifolds. RMPflow generalizes operational space control (Khatib, 1987) to model abstract and non-Euclidean behaviors and can be combined with motion planning, such as the work from Ratliff, Toussaint, and Schaal (2015a). Theoretically, we show RMPflow is Lyapunov-stable and has a coordinate-free expression in tangent bundles. This means that policy of one robot can be stably transferred to another robot without learning, given enough degrees of freedom. These properties are proved through defining a new class of differential equations called Geometric Dynamical Systems (GDSs), which generalize the traditional Simple Mechanical Systems (SMSs) to have inertia matrices that are *velocity-dependent*. The incorporation of velocity information in inertia matrices in RMPflow is a critical step to increasing the policy class' expressivity; e.g., it allows RMPflow to generate natural and fast collision avoidance behaviors which previous methods fail.

In Chapter 12, we formally show that RMPflow is fully differentiable and therefore can be viewed as a policy class in policy learning (Mukadam et al., 2019). In addition, we propose a variation of RMPflow, called RMPfusion, where extra weight functions are introduced onto the edges of the tree data structure of RMPflow. This modification acts

as two roles: on the one hand, it increases the programming flexibility when one hand-specifies task-space RMP policies; on the other hand, it can be used to more finely control the policy fusion process. We show that, like other parameters in RMPflow, these weights can also be learned through back-propagation. Furthermore, we prove that RMPfusion provides the same type of stability guarantees as RMPflow, so long as the weight functions satisfy minor restrictions (such as being non-negative).

1.4.2 Combining Policies with Control Lyapunov Functions

Nevertheless, the above advancement in policy fusion made by RMPflow relies on the assumption that the task-space policies are GDSs in order to guarantee stability. Although GDSs are already more general than SMSs that form the basis of operational space control, they are still a very specific case of general second-order dynamical systems, and asking engineers to be familiar these new GDSs increases the learning curve of using RMPflow.

Fortunately, in Chapter 13, we show that the policy fusion procedure of RMPflow actually works beyond GDSs: RMPflow remains stable for all task-space policies that are stable with respect to a family of Control Lyapunonv Functions (CLFs) (Li et al., 2019b). In other words, we can use RMPflow to combine a much larger class of policies, without changing the algorithm. Leveraging this finding, we design an efficient CLF-based computational framework that can combine arbitrary subtask policies provided by users into a globally stable policy. Compared with the original usage of RMPflow above, this new framework gives users the flexibility to incorporate design heuristics through nominal controllers for the subtasks.

In summary, RMPflow (and its variation RMPfusion) leverage a tree-decomposition of the problem and effectively represents a policy class that is Lyapunov-stable and yet is expressive enough to contain policies that are known to work well in practice. These properties make RMPflow an ideal candidate for policy optimization with requirements of safety and interpretability.

1.5 Outline

The following chapters of this thesis are organized as follows. We partition the overall thesis into three parts:

- Part I: Policy Optimization I: Imitation
- Part II: Policy Optimization II: Abstraction
- Part III: Structral Policy Fusion

In Part I, we first introduce the setup and preliminaries of policy optimization and imitation learning (IL) in a tutorial style, respectively, in Chapter 2 and Chapter 3. Then we show in Chapter 4 that IL can be used as a tool to speed up policy optimization with real-world experiments in autonomous driving. In Chapter 5, we provide a simple solution to deal with suboptimal expert policies. Inspired by these encouraging results of online IL, we dive deeper into the theoretical properties of online IL in Chapter 6. Finally, we show in Chapter 7 how these new insights can be leveraged to further accelerate online IL via the concept of predictive models.

Part II of this thesis the generalizes the new theoretical findings of IL presented in Part I to a larger class of problems. In Chapter 8, we establish a refinement of existing online learning setups, called Continuous Online Learning (COL), to formally consider the regularity of loss functions across different rounds, an important observation made in Chapter 6. Based on this new formulation, in Chapter 9, we provide a reduction from the saddle-point formulation of RL to COL, by which efficient policy optimization algorithms with strong theoretical guarantees can be systematically designed. Lastly, we describe the meta algorithm PICCOLO in Chapter 10, which generalizes the results of Chapter 7 and provides a principled way to leverage the predictable information resulted from the regularity of online losses to speed up learning.

Part III complements the first two parts of this thesis by providing a class of second-order policies with provable stability guarantees. In other words, under proper assumptions, these structural policies can be used in conjunction with the learning techniques described in Parts I and II, so that even policies generated in the premature learning phase are guaranteed to be Lyapunov stable. We first describe the general policy fusion algorithm, RMPflow, and its geometric and stability properties in Chapter 11. Next in Chapter 12, we provide an extension, RMPfusion, to further increase its parameterization flexibility. Finally, in Chapter 13, we re-establish and extend the stability results of RMPflow in Chapter 11 through a rigorous Control Lyapunov Function (CLF) treatment, and provide a new Lyapunov stable framework for combining arbitrary user policies.

At the very end, in Chapter 14, we revisit the question of whether better theories can help bridge the reality gap. The short answer is yes. Nonetheless, the insights learned here also open the door to many new, exciting research directions.

Part I

Policy Optimization I: Imitation

ABSTRACT

In this part of the thesis, we discuss how imitation learning (IL) can be used for efficient policy optimization (Cheng and Boots, 2018; Cheng et al., 2018a, 2019b; Pan et al., 2019; Pan et al., 2018). First, in Chapter 2, we define a general policy optimization problem and set up the notation for the remaining chapters. This chapter is written in a tutorial style with an aim to provide background knowledge in reinforcement learning (RL) to unfamiliar readers. In Chapter 3, we discuss domain knowledge available in practical policy optimization problems, and show how imitation learning (IL) can be used as a interface to express this prior knowledge. We compare different approaches to IL and theoretically analyze their properties. Later in Chapter 4 we use these insights to design a sample-efficient IL system for off-road high-speed autonomous driving.

With these encouraging results, we further study the theoretical properties of online IL. In Chapter 5, we address the issue of suboptimal expert policies through a simple randomized framework that combines IL and RL. In Chapter 6, we finely characterize interesting phenomena about the convergence of a popular IL algorithm AGGREVATTE, proving that whether it converges in the last iterate or not is purely problem dependent. This insight supports the usage of common heuristics in online IL, as we show that these tricks effectively act as regularization to stabilize the problem. Inspired by these findings, we design two new online IL algorithms (MOBIL) in Chapter 7. These algorithms learn predictive models of future gradients online, and can provably accelerate the convergence of existing IL methods in literature up to an order.

NOTATION

Table 1.1: Notation for Policy Optimization

Symbol	Definition
$\mathcal{S}$	the state space
$\mathcal{A}$	the action space
$\pi(a\|s)$	the distribution of a policy π, where $s \in \mathcal{S}$ and $a \in \mathcal{A}$
$\pi^\star(a\|s)$	the expert policy
γ	the discount factor
T	the problem horizon
t	the time step
$r(s,a)$	the reward function
$c(s,a)$	the cost function
$\mathcal{P}(s'\|s,a)$	the transition dynamics
$p(s)$	the initial state distribution
$V^\pi(s)$	the value function of a policy π, where $s \in \mathcal{S}$
$Q^\pi(s,a)$	the Q-function of a policy π, where $s \in \mathcal{S}$ and $a \in \mathcal{A}$
$V^\pi(p), J(\pi)$	the return (i.e. the expected accumulated cost/reward) of a policy π
$\bar{V}^\pi(p), \bar{J}(\pi)$	the average return of a policy π
$d_t^\pi(s)$	the state distribution of a policy π at time t
$d^\pi(s)$	the average state distribution of a policy π

Table 1.2: Notation for Online Learning

Symbol	Definition
n	the round number
N	the total number of rounds
$\mathcal{X}$	the decision set
l_n	the loss in round n
$\mathrm{regret}_N(\mathcal{X})$	the static regret w.r.t. to the adversarial comparator in $\mathcal{X}$ after N rounds
$\mathrm{regret}_N^s(\mathcal{X})$	the same as $\mathrm{regret}_N(\mathcal{X})$
$\mathrm{regret}_N^d(\mathcal{X})$	the dynamic regret w.r.t. to the adversarial comparator in $\mathcal{X}$ after N rounds

CHAPTER 2
POLICY OPTIMIZATION

In this chapter, we provide a tutorial of reinforcement learning (RL) from the perspective of policy optimization. We will use this chance to set up the notation for the following chapters. While the contents here are not new, our goal is to summarize these preliminaries in a coherent, concise manner and highlight important properties. We invite the readers to refer to, e.g., (Puterman, 2014) for further details.

2.1 Setup

We consider sequential-decision making problems in discrete-time Markov decision processes (MDPs). Let $\mathcal{S}$ denote the state space, $\mathcal{A}$ denote the action space, and $\mathcal{P}(s'|s,a)$ denote the (stochastic) transition dynamics, for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. We allow the state and actions spaces to be either discrete or continuous. To make writing compact, we will embed time information into the state definition when needed (e.g. in finite-horizon applications); that is, one can think of $\mathcal{S}$ as $\bar{\mathcal{S}} \times [0, \infty)$, where $\bar{\mathcal{S}}$ is some basic state space, and $\mathcal{P}(s'|s,a)$ as a non-stationary function for elements in $\bar{\mathcal{S}}$. For simplicity, we suppose the same action space $\mathcal{A}$ is shared across different states; the following results can be easily extended to the state-dependent case.

In policy optimization, the goal is to find a policy $\pi(a|s)$ (a conditional distribution of action $a \in \mathcal{A}$ given state $s \in \mathcal{S}$) inside a compact policy class Π with good performance. In this thesis, we will consider both the reward and the cost formulations, whichever is more appropriate for a given application, where we use $r(s,a)$ and $c(s,a)$ to denote the reward and the cost functions, respectively. Following the standard convention, we measure the performance of a policy π in terms of $V^\pi(p)$, the *expected accumulated reward/cost* (or simply the *return*) when π is executed in the MDP starting from some initial state

distribution $p(s_0)$. In the cost formulation, we would often write $J(\pi) \coloneqq V^\pi(p)$, which is known as the cost-to-go in the control literature. Since cost and reward are essentially interchangeable, we will focus only on the reward case in this tutorial chapter.

We will study both the finite-horizon and the discounted infinite horizon cases:

- Finite-horizon problems with horizon $T < \infty$:

$$V^\pi(p) \coloneqq \mathbb{E}_{s_0,a_0,s_1,\cdots\sim\rho^\pi(p)}\left[\sum_{t=0}^{T-1} r(s_t,a_t)\right] \tag{2.1}$$

- Infinite-horizon problems with discount $\gamma \in [0,1)$:

$$V^\pi(p) = \mathbb{E}_{s_0,a_0,s_1,\cdots\sim\rho^\pi(p)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\right] \tag{2.2}$$

where $\rho^\pi(p)$ is the trajectory distribution of running the policy π from the initial distribution p. Likewise we write $\rho^\pi(s)$ if the trajectory starts from a state $s \in \mathcal{S}$. Note again that, because we allow the state to encode time information, the policies considered here can effectively be non-stationary if needed.

While $V^\pi(p)$ is the performance of interest, we will more often work with its average version, which we denote as $\bar{V}^\pi(p)$, in order to design algorithms that are agnostic the exact setup. Specifically, the *average return* $\bar{V}^\pi(p)$ is related to the return $V^\pi(p)$ by $\bar{V}^\pi(p) = \frac{1}{T}V^\pi(p)$ for the T-horizon problem, or $\bar{V}^\pi(p) = (1-\gamma)V^\pi(p)$ for the γ-discounted infinite horizon problem. The average return can alternatively be defined from the perspective of the average state distribution. Let $d_t^\pi(s)$ denote the state distribution at time t of running the policy π from the initial distribution p. We define the *average state distribution* as

$$d^\pi \coloneqq \frac{1}{T}\sum_{t=0}^{T-1} d_t^\pi(s) \qquad \text{or} \qquad d^\pi \coloneqq (1-\gamma)\sum_{t=0}^{\infty} \gamma^t d_t^\pi(s) \tag{2.3}$$

for the T-horizon and the γ-discounted infinite horizon problems, respectively. Then the

average return, for both settings, can be defined using the same expression as

$$\bar{V}^{\pi}(p) \coloneqq \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi|s}[r(s,a)]. \tag{2.4}$$

Likewise we can define $\bar{J}(\pi)$. For convenience, we will often omit writing the random variable in expectations; for example, we may write (2.4) simply as $\bar{V}^{\pi}(p) = \mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[r]$.

From the above construction, we see that the factor $\frac{1}{1-\gamma}$ plays the role of the effective horizon in a γ-discounted infinite horizon problem, or the factor $1 - \frac{1}{T}$ is the effective discount factor in a T-horizon problem. Therefore, core mathematical properties of finite horizon problems and discounted infinite horizon problems are essentially the same, though algebraic differences are necessary in writing.

We also notice that, by the definition in (2.4), as either $\gamma \to 1$ or $T \to \infty$, the average return $\bar{V}^{\pi}(p)$ converges to the objective of the undiscounted infinite horizon MDP problems. In the limit, the average state distribution d^{π} defined in (2.3) is known as the limiting average state distribution, which is the same as the stationary distribution of the MDP under π when one exists (Puterman, 2014). Because the following chapters are based on the average setup in (2.4), most of our results also apply to the undiscounted infinite horizon case.

2.2 Objective and Assumptions

Based on the definition of average return in (2.4), a general policy optimization problem in RL can be written compactly as

$$\max_{\pi\in\Pi} \bar{V}^{\pi}(p) = \max_{\pi\in\Pi} \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi|s}[r(s,a)]. \tag{2.5}$$

The main assumption we have imposed so far is that the policy class Π is compact and the MDP is discrete-time.[1] We should remark that, for the purpose of policy optimization,

[1] Similar constructions could be extended to continuous time under technical regularity assumptions.

assuming the state observable does not lose generality. Suppose originally we wish to optimize some reactive policy $\pi'(a|o)$ in a partially observable MDP with an observation space $\mathcal{O}$, where $o \in \mathcal{O}$. We can equivalently view this problem as optimizing the policy $\pi(a|s) = \mathbb{E}_{o|s}[\pi'(a|o)]$ in the fully observable MDP. For history-dependent policies, similar ideas of transformation can be applied, as a state by definition captures all the information. In other words, partial observability simply means that we have restrictions on the structure of the policy class Π.

We have not made assumptions on the initial state distribution p and the transition dynamics $\mathcal{P}$; they can be unknown, and $\mathcal{P}$ can be stochastic. We also have not yet made any restrictions on the reward r and the cost c functions. Therefore, the problem described in (2.5) is fairly general; the only way for the learner to improve the policy π is to directly interact with the unknown MDP to gather information.

To make learning feasible, we will additionally assume that the problem can be reset by querying the (unknown) initial state distribution, so that the learner would not run into disastrous and unrecoverable situations (like getting stuck in a sub-optimal absorbing state). We call this setting *episodic policy optimization*, where the data are collected in terms of trajectories. We will make this assumption, if not stronger, throughout the following chapters.

Later we will also add extra structural assumptions on the generic setup (2.5), as often prior knowledge about the MDP problem at hand is available. While we will give the specific insights, we recall typical assumptions used in the literature.

- In a typical RL problem, the reward is usually bounded, e.g., in the range $[0, 1]$.
- When the state and action spaces are continuous, (local) Lipschitz properties of the reward/cost and the transition dynamics are likely to hold.
- In control applications, the cost function is hand-specified and therefore known (which is usually continuous though not necessarily bounded), and partial knowledge about

the transition dynamics and the initial state distribution is not uncommon.

- In imitation learning (IL), the prior knowledge about the problem is represented in the form of an expert policy, although the reward/cost in this case can sometimes be completely unknown and cannot be queried at all.

2.3 A General Performance Difference Lemma

Let us discuss some useful properties that are induced by the Markovian structure of the MDP and the additive structure of the performance index in (2.4). Working with these properties are often more convenient than directly with the algebra.

First, using (2.4), we define the *value function* of the policy π at state $s \in \mathcal{S}$ as $V^\pi(s)$, i.e. it is the return when the initial distribution is a delta distribution centered at s.[2] Using $V^\pi(s)$, we can define the Q-functions with respect to the policy π: with[3] $\gamma \in [0, 1]$,

$$Q^\pi(s, a) \coloneqq r(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}|s,a}[V^\pi(s')] \tag{2.6}$$

It is easy to verify that they are related through $V^\pi(s) = \mathbb{E}_{a \sim \pi|s}[Q^\pi(s, a)]$. In other words, we have the recursive equation,

$$V^\pi(s) = \mathbb{E}_{s \sim \pi|s}\left[r(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}|s,a}[V^\pi(s')]\right] \tag{2.7}$$

For the optimal policy π^*, its value function additionally satisfies the Bellman equation (Bellman, 1954)

$$V^\pi(s) = \max_{a \in \mathcal{A}} r(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}|s,a}[V^\pi(s')]. \tag{2.8}$$

Using the value function and the Q-function, we can define the advantage function with

[2] For finite-horizon problems, the sum starts from the associated time step of s and stops when the end of horizon is reached.

[3] For the finite-horizon problem, it holds with $\gamma = 1$ as the state includes time information.

respect to the policy π:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s), \tag{2.9}$$

which measures the benefits of switching to an action $a \in \mathcal{A}$ from using the policy π at state s (assuming π is used afterwards). The advantage function is a very convenient quantity to encapsulate a policy property, especially, when it is combined with the performance difference lemma below due to Kakade and Langford (2002).

Lemma 2.3.1 (Performance Difference Lemma)**.** (Kakade and Langford, 2002) *Let π and π' be two policies. Then*

$$\bar{V}^{\pi}(p) = \bar{V}^{\pi'}(p) + \mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[A^{\pi'}] = \bar{V}^{\pi'}(p) + \mathbb{E}_{d^{\pi}}\left[\mathbb{E}_{\pi}[Q^{\pi'}] - \mathbb{E}_{\pi'}[Q^{\pi'}]\right]$$

We will prove a more general version of the above the performance difference lemma. Before that let us discuss some of its implications. An immediate consequence of the performance difference lemma is the policy gradient theorem.

Corollary 2.3.1. $\mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[A^{\pi}] = 0$

Corollary 2.3.2 (Policy Gradient)**.** *If π is parameterized by θ.* $\nabla_{\theta}\bar{V}^{\pi}(p) = \mathbb{E}_{d^{\pi}}(\nabla_{\theta}\mathbb{E}_{\pi})[A^{\pi}]$

Proof. Observe $\nabla_{\theta}\bar{V}^{\pi}(p) = (\nabla_{\theta}\mathbb{E}_{d^{\pi}})\mathbb{E}_{\pi}[A^{\pi'}] + \mathbb{E}_{d^{\pi}}(\nabla_{\theta}\mathbb{E}_{\pi})[A^{\pi'}]$ and $\mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[A^{\pi}] = 0$. Then set $\pi' = \pi$. We have the policy gradient theorem. ■

In practice, $\mathbb{E}_{d^{\pi}}(\nabla_{\theta}\mathbb{E}_{\pi})[A^{\pi}]$ is approximated by sampling. This can be done by using the likelihood-ratio method based on the equality $\mathbb{E}_{d^{\pi}}(\nabla_{\theta}\mathbb{E}_{\pi})[A^{\pi}] = \mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[(\nabla_{\theta}\log\pi)A^{\pi}]$.

The performance difference lemma is also closely related to the idea of reward reshaping (Ng, Harada, and Russell, 1999) (i.e. modifying the reward function while not changing the objective). Now we give a generalized version that subsumes both cases discussed above and has a simpler proof.

Lemma 2.3.2 (General Performance Difference Lemma)**.** *Let π be any policy and f be any function, which can be history dependent. For the γ-discounted infinite-horizon problem,*

$$\bar{V}^{\pi}(p) = (1-\gamma)\mathbb{E}_p[f] + \mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[A^f]$$

where $A^f(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}|s,a}[f(s')] - f(s)$.

Proof. We prove the statement by a simple telescoping sum.

$$\begin{aligned}
V^{\pi}(p) &= \mathbb{E}_{s_0,a_0,s_1,\cdots \sim \rho^{\pi}(p)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \\
&= \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim d_t^{\pi}}\mathbb{E}_{a_t\sim\pi|s_t}[r(s_t,a_t)] + \mathbb{E}_{s_t\sim d_t^{\pi}}[f(s_t)] - \mathbb{E}_{s_t\sim d_t^{\pi}}[f(s_t)] \\
&= \mathbb{E}_{s_t\sim d_0^{\pi}}[f(s_t)] + \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim d_t^{\pi}}\mathbb{E}_{a_t\sim\pi|s_t}[r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\mathcal{P}|s_t,a_t}[f(s_{t+1})] - f(s_t)] \\
&= \mathbb{E}_p[f] + \frac{1}{1-\gamma}\mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[A^f]
\end{aligned}$$

■

Corollary 2.3.3. *Let π be any policy and f be any function which can be history dependent but satisfies that $f(s_t) = 0$ for s_t at time step $t \geq T$. For the T-horizon problem, it holds*

$$\bar{V}^{\pi}(p) = \frac{1}{T}\mathbb{E}_p[f] + \mathbb{E}_{d^{\pi}}\mathbb{E}_{\pi}[A^f]$$

where $A^f(s,a) = r(s,a) + \mathbb{E}_{s' \sim \mathcal{P}|s,a}[f(s')] - f(s)$.

Lemma 2.3.2 supports a non-stationary, history-dependent functional comparator that is not necessarily a value function. In comparison, the reward reshaping lemma (Ng, Harada, and Russell, 1999) assumes that the comparator is fixed function of the basic state, and the performance difference lemma in (Kakade and Langford, 2002) assumes that the comparator is the value function of some policy. In addition, we see that, in Lemma 2.3.2, the comparator $f(s_t)$ for s_t at time t can be constructed online, depending on the history $h_t = s_0, a_0, s_1, \ldots, s_t$. This lemma can potentially be helpful to designing trajectory-wise

control variates for variance reduction, although we will be mainly using the basic version of the performance difference lemma in the later chapters.

CHAPTER 3
IMITATION LEARNING

3.1 Introduction

Understanding policy optimization for sequential decision making in uncertain environment is fundamental to robotics. As we show in Chapter 2, this problem can be mathematically formulated as a reinforcement learning (RL) problem. However, learning a policy from scratch with a standard RL algorithm (like the policy gradient method (Williams, 1992)) faces many practical challenges, such as high-variance, sparse feedback signals, and unsafe explorations. As a consequence, these RL techniques are often considered to be too data inefficient and unsuitable for robotics applications, because collecting real-world data is both expensive and time-consuming.

Fortunately, we usually have some prior knowledge about the policy optimization problem at hand. For example, even though we do not know the optimal feedback controller for a robot, we usually already have a set of hand-tuned PID controllers that can work up to certain degree. In other situations, we may have access to human experts who can perform similar tasks, or information of the dynamics in the sequential decision process. Therefore, in practice, we rarely are solving the most generic policy optimization in RL.

A powerful and systematic way to leverage this prior information is imitation learning (IL). IL abstracts this domain knowledge as *expert policies*. In the above examples, the PID controllers and the human demonstrators are directly in the form an expert policy, which can provide desired or non-trivial actions for different situations. For the knowledge of dynamics, we can express it as an expert policy, e.g., through designing an optimal controller with respect to the dynamics model. Therefore, the concept of expert policies can be used as a unified channel to express our understanding about the problem.

IL avoids the aforementioned issues in RL by optimizing policies through *surrogate problems* that are defined by demonstrations of the expert policy. As we will show later, these surrogate problems can be a supervised learning problem (Pomerleau, 1989), an online learning problem (Ross, Gordon, and Bagnell, 2011), or even another (perhaps simpler) RL problem (Peng et al., 2018; Schroecker and Isbell, 2017). For example, we can set up an optimization problem of which the objective function is some distance measure between the learner policy and the expert policy, and intuitively minimizing this objective function can drive the learner policy to behave similarly as the expert policy. The hope is that these surrogate problems have better optimization properties (like smaller variance and denser costs) than the original RL problem of interest, while being able to provide meaningful directions for policy improvement.

But what properties constitute a good surrogate problem for IL? A good surrogate problem in IL should be easy to solve and in the meanwhile can lead to a policy that performs well in the original RL problem. However, these two requirements often lead to a trade-off. For example, training a policy in a supervised learning fashion, with samples collected by the expert policy, is perhaps the simplest surrogate problem in IL (this scheme is known as batch IL or behavior cloning). However, performing well on such a surrogate problem, in general, does not necessarily translate into good performance in the original RL problem. The main reason for the performance gap is known as the *covariate shift problem*: the learner policy may visit a set of states different from the ones visited by the expert policy in the training set. Therefore, when the learner runs its policy and ends up in states that cannot find resemblance in the training set, there is no guarantee that the learner can still take meaningful actions. Overall the rule of thumb is that surrogate problems easier to optimize usually need to pay for larger covariate-shift effects. For IL to be effective, designing a balanced surrogate problem for the application domain is an important topic.

In this chapter, we give a concise introduction to IL and discuss the strengths and weakness of different approaches. Particularly, we will focus on problems with continuous ac-

tion spaces, as our ultimate goal is to find a suitable framework for robotics applications. Our presentation is motivated by the realizations that the connection between online IL and DAGGER-like algorithms (Ross, Gordon, and Bagnell, 2011) has not been formally introduced in continuous domains,[1] and that the original derivation is more convoluted and does not convey important structural properties. Here we simplify the derivation of Ross, Gordon, and Bagnell (2011) into a compact tutorial. Some materials here are based on the talk I presented in (Cheng, 2018) and our papers published as (Pan et al., 2019; Pan et al., 2018)

3.2 Problem Setup

Let us quickly refresh the problem setup. We consider a discrete-time, continuous-valued sequential-decision making problem with a finite horizon T. Let $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ be the state, action, and observation spaces. Our goal here is to find a stationary, reactive policy[2] $\pi : \mathcal{O} \mapsto \mathcal{A}$ inside a policy class Π such that π has a low accumulated cost:

$$\min_{\pi} J(\pi), \quad J(\pi) \coloneqq \mathbb{E}_{s_0,o_0,a_0,s_1,\ldots,a_{T-1}\sim\rho^{\pi}(p)} \left[\sum_{t=0}^{T-1} c(s,a) \right], \tag{3.1}$$

in which $s \in \mathcal{S}$, $o \in \mathcal{O}$, $a \in \mathcal{A}$, $c(s,a)$ is the instantaneous cost, and $\rho^{\pi}(p)$ is the distribution of trajectory generated by running the policy π starting from a fixed initial state distribution p at time 0. Note that $\mathcal{S}, \mathcal{O}$ contain time information (see Chapter 2).

While the policy π above only have access to the observations, we recall from Chapter 2 that the partial observability can be handled through considering the effective policy class in analysis. In other words, we can embed the partial observability into the policy structure and frame the above problem equivalently as policy optimization in the MDP with this structured policy class. Therefore, in the following, we will abuse the notation: identify

[1] Before our conference paper (Pan et al., 2018) was published, DAGGER had only been used heuristically in these domains (Ross et al., 2013; Zhang and Cho, 2016)

[2] While we focus on reactive policies, the same derivations apply to history-dependent policies.

$\pi(a|s) = \mathbb{E}_{o|s}[\pi(a|o)]$ and modify Π accordingly.

The below discussions will also use the concept of Q-function and value function. Because our state/observation definition contains time information, when using $Q^\pi(s,a)$ to denote the Q-function value for some state s and action a at time t, it means that

$$Q^\pi(s,a) = \mathbb{E}_{s,o,a,\ldots,a_{T-1}\sim\rho^\pi(s)|a=a}\left[\sum_{\tau=t}^{t+T-1} c(s_\tau, a_\tau)\right]$$

Similarly, $V^\pi(s) = \mathbb{E}_{a\sim\pi|s}[Q^\pi(s,a)]$ is the associated non-stationary value function.

3.3 Goal of Imitation Learning

Directly optimizing (3.1) is challenging. Especially when the agent is a physical robot, model-free RL techniques become intolerably sample inefficient and have the risk of permanently damaging the robot when deploying a partially-optimized policy in exploration. Considering these limitations, we consider to solve for policy π by IL. We assume the access to an oracle policy or *expert* $\pi^\star$ to generate demonstrations during the training phase. This expert can use resources that are unavailable in the testing phase, like additional sensors and computation. The goal of IL is to quickly find a policy π for the learner such that its performance $J(\pi)$ is close to $J(\pi^\star)$ of the expert policy $\pi^\star$.

3.3.1 Performance Difference

To this end, we need a way to express the performance difference $J(\pi) - J(\pi^\star)$. A useful tool is the performance difference lemma below, which was due to Kakade and Langford (2002). Define $\bar{J}(\pi) \coloneqq \frac{1}{T}J(\pi)$. We will write the results instead in terms of the average setup with $\bar{J}(\pi)$, so that our results can be extended to other settings (see Chapter 2).

Lemma 3.3.1. *Let $d^\pi(s) = \frac{1}{T}\sum_{t=0}^{T-1} d_t^\pi(s)$ be the average state distribution,*[3] *where d_t^π is the distribution of state at time t when running the policy π. Let π and π' be two arbitrary policies. Then it holds that $\bar{J}(\pi) = \bar{J}(\pi') + \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}[A^{\pi'}(s,a)]$, where $A^{\pi'}(s,a) =$*

$Q^{\pi'}(s,a) - V^{\pi'}(s)$ *is the (dis)advantage function with respect to* π'.

Lemma 3.3.1 gives a closed-form expression of the performance difference based on structural properties of MDPs. Precisely, it can be read in two ways:

$$\bar{J}(\pi) - \bar{J}(\pi^\star) = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}[A^{\pi^\star}(s,a)] \tag{3.2}$$

$$= -\mathbb{E}_{s\sim d^{\pi^\star}}\mathbb{E}_{a\sim\pi^\star|s}[A^{\pi}(s,a)] \tag{3.3}$$

giving a duality relationship of the performance difference. First, (3.2) says that the performance difference is equivalent to the degree the learner policy π is better than the expert policy $\pi^\star$ in expectation over the states visited by the learner, because the advantage function $A^{\pi^\star}(s,a)$ in essence measures how an action a is better than the expert policy $\pi^\star$ at state s. Alternatively, (3.3) says that the performance difference is equivalent to the degree the expert policy $\pi^\star$ is worse than the learner policy π in expectation over the states the expert visits. In short, these two perspectives show that the performance difference can be upper bounded in terms of either the state-action distribution of the learner or that of the expert. We will show that the choice of perspective, (3.2) or (3.3), is the bifurcation point that leads to the online approach or the batch approach to IL.

To derive upper bounds of (3.2) and (3.3), let us review a basic statistical distance, called the transportation distance (Kontorovich and Raginsky, 2017). We will use this distance to derive, e.g., upper bounds of (3.2) and (3.3). For two probability distributions p and q on a metric space $\mathcal{M}$ with metric m, the *transportation distance* is defined as

$$D_W(p,q) = \inf_{g\in\Gamma(p,q)} \mathbb{E}_{x,y\sim g}[m(x,y)] \tag{3.4}$$

where Γ denotes the family of distributions whose marginals are p and q. It can be shown by the Kantorovich-Rubinstein theorem that the above definitions has an equivalent dual

[3] $d^\pi(s)$ is an unnormalized time-state distribution of the time-state distribution $\frac{1}{T}d^\pi(s)$. One can verify that $\sum_{t=1}^{T}\int_{s\in\mathcal{S}}\frac{1}{T}d^\pi(s) = 1$.

representation (also known as the Wasserstein distance):

$$D_W(p,q) = \sup_{f:\text{Lip}(f(\cdot))\leq 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)] \tag{3.5}$$

where $\text{Lip}(f(\cdot))$ denotes the Lipschitz constant of a function f with respect to the metric m. Therefore, it follows that any L-Lipschitz function f in the metric space $\mathcal{M}$ satisfies

$$\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)] \leq LD_W(p,q)$$

The transportation distance is a flexible tool for upper bounding the difference between expectations, as we can choose the metric m according to the geometry of the random variables to compensate for the effects of dimensionality. For example, when the random variables are of discrete values, we can choose the metric as the indicator function and the transportation distance is then equal to the total variation distance. Our interest here concerns continuous-valued random variables in some normed space with norm $\|\cdot\|$. Naturally, we can choose the metric as the distance metric induced by the norm $\|\cdot\|$, i.e. $d(x,y) = \|x-y\|$. Finally, we note that the below results apply to also other statistical distances that further upper bound the transportation distance, like the KL-divergence.

3.4 Different Approaches to Imitation Learning

3.4.1 Online Imitation Learning

We first present the objective function for the online learning approach to IL. Let $C^{\pi^\star} = \sup_{s\in\mathcal{S}} \text{Lip}\left(Q^{\pi^\star}(s,\cdot)\right)$ be the uniform Lipschitz constant, which describes the regularity of expected behaviors. We achieve this by upper bounding the performance difference

using (3.2) as follows

$$\begin{aligned}\bar{J}(\pi) - \bar{J}(\pi^\star) &= \mathbb{E}_{s\sim d^\pi}\left[\mathbb{E}_{a\sim\pi|s}[Q^{\pi^\star}(s,a)] - \mathbb{E}_{a^\star\sim\pi^\star|s}[Q^{\pi^\star}(s,a^\star)]\right] \\ &\leq C^{\pi^\star}\mathbb{E}_{s\sim d^\pi}\left[D_W(\pi,\pi^\star)\right] \\ &\leq C^{\pi^\star}\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}\mathbb{E}_{a^\star\sim\pi^\star|s}[\|a-a^\star\|], \end{aligned} \tag{3.6}$$

where the first equality is simply (3.2) but with the advantage function replaced with its definition $A^{\pi^\star}(s,a) = Q^{\pi^\star}(s,a) - \mathbb{E}_{a^\star\sim\pi^\star|s}[Q^{\pi^\star}(s,a^\star)]$, and the two inequalities are due to (3.5) and (3.4), respectively. Define $\hat{c}(s,a) = \mathbb{E}_{a^\star\sim\pi^\star|s}[\|a-a^\star\|]$. Thus, to make π perform as well as $\pi^\star$, we can minimize the upper bound in (3.6), i.e.

$$\min_\pi \mathbb{E}_{s_0,a_0,\ldots,a_{T-1}\sim\rho^\pi(p)}\left[\sum_{t=0}^{T-1}\hat{c}(s_t,a_t)\right]. \tag{3.7}$$

This new surrogate loss in (3.7) is the objective of the *online* IL. Notice that this is still an RL problem because the trajectory distribution in (3.7) still depends on π. One can choose to solve (3.7) with pure RL techniques (e.g. policy gradient methods); however, while this new problem might be simpler than the original one in (3.1) (e.g. (3.7) has denser cost functions than (3.1)), tackling it directly could still be sample inefficient due to the necessity of back-propagating information through trajectories. To circumvent this difficulty, the online learning approach to IL (Cheng et al., 2019b,d; Ross and Bagnell, 2014; Ross, Gordon, and Bagnell, 2011) leverages the structural property of cost function $\hat{c}$ in (3.7) and relies on a reduction from (3.7) to online learning problems (Shalev-Shwartz, 2012) to optimize policies. As a result, back-propagating through trajectories is no longer necessary, and provable performance guarantees can be achieved (Cheng et al., 2018a). We will further discuss about this structural property of $\hat{c}$ in Chapter 6. For now, we highlight that this reduction works, because $\hat{c}(s,a)$ behaves like a loss function in supervised learning, i.e., given *any* s there is some a (namely the action taken by the expert) such that $\hat{c}(s,a)$ is close

to zero.

Let us illustrate this idea by the classic online IL algorithm, DAGGER (Data Aggregation) (Ross, Gordon, and Bagnell, 2011), which reduces (3.7) to a sequence of supervised learning problems: Let $\mathcal{D}$ be the training data of pairs of state and action. DAGGER initializes $\mathcal{D}$ with samples gathered by running $\pi^\star$. Then, in the nth iteration, it trains π_n by supervised learning,

$$\pi_n = \arg\min_{\pi \in \Pi} \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\pi|s}[\hat{c}(s,a)], \tag{3.8}$$

where the subscript $\mathcal{D}$ denotes empirical data distribution. Next it runs π_n to collect more data, which is then added into $\mathcal{D}$ to train π_{n+1}. The procedure is repeated for $O(T)$ iterations and the best policy, in terms of (3.7), is returned. Suppose the policy is linearly parametrized. When the online loss $\mathbb{E}_{s\sim d^{\pi_n}} \mathbb{E}_{a\sim\pi|s}[\hat{c}(s,a)]$ defined by the policy π_n in the nth iteration is strongly convex, running DAGGER to solve (3.7) finds a policy π with performance $J(\pi) \leq J(\pi^\star) + O(TC^{\pi^\star})$ (recall that $J(\pi) = T\bar{J}(\pi)$). We note here the instantaneous cost $\hat{c}(s,\cdot)$ can be selected to be any suitable norm according the problem's property since norms in finite dimensional space are equivalent.

3.4.2 Batch Imitation Learning

The batch approach to IL (Pomerleau, 1989) takes a different viewpoint of performance difference. Using the other equality in (3.3), we can derive another upper bound and use it to construct a different surrogate problem: define $\tilde{c}^\pi(s^\star, a^\star) = \mathbb{E}_{a\sim\pi|s^\star}[\|a - a^\star\|]$ and $C^\pi(s^\star) = \text{Lip}(Q^\pi(s^\star,\cdot))$, then we can write

$$\begin{aligned}\bar{J}(\pi) - \bar{J}(\pi^\star) &= \mathbb{E}_{s^\star\sim d^{\pi^\star}}\left[\mathbb{E}_{a\sim\pi|s^\star}[Q^\pi(s^\star,a)] - \mathbb{E}_{a^\star\sim\pi^\star|s^\star}[Q^\pi(s^\star,a^\star)]\right] \\ &\leq \mathbb{E}_{s^\star\sim d^{\pi^\star}} \mathbb{E}_{a^\star\sim\pi^\star|s^\star}\left[C^\pi(s^\star)\tilde{c}^\pi(s^\star,a^\star)\right]. \end{aligned} \tag{3.9}$$

where the derivation is similar to the one in (3.6) but the first equality is instead based on (3.3). The problem of minimizing the upper-bound (3.9) is called the *batch* IL prob-

lem (Bojarski et al., 2017; Rausch et al., 2017) and can be written equivalently as:

$$\min_{\pi} \mathbb{E}_{s_0^\star, a_0^\star, \ldots, a_{T-1}^\star \sim \rho^{\pi^\star}(p)} \left[\sum_{t=1}^{T} \tilde{c}^{\pi}(s^\star, a^\star) \right], \tag{3.10}$$

In contrast to the surrogate problem in online IL (3.7), batch IL reduces to a supervised learning problem, because the expectation is defined by a fixed policy $\pi^\star$.

3.4.3 Imitation Learning without Demonstrations

The two approaches require getting action demonstrations from the expert policy. When this is not feasible, IL is possible when the instantaneous cost function does not depend on the action, i.e. $c(s, a) = c(s)$. Suppose $c(\cdot)$ is C^c-Lipschitz. In this case, we can write the performance difference as

$$\begin{aligned}
\bar{J}(\pi) - \bar{J}(\pi^\star) &= \mathbb{E}_{s\sim d^\pi} \mathbb{E}_{a\sim\pi|s}[c(s,a)] - \mathbb{E}_{s^\star\sim d^{\pi^\star}} \mathbb{E}_{a^\star\sim\pi^\star|s^\star}[c(s^\star, a^\star)] \\
&= \mathbb{E}_{s\sim d^\pi}[c(s)] - \mathbb{E}_{s^\star\sim d^{\pi^\star}}[c(s^\star)] \\
&\leq C^c D_W(d^\pi, d^{\pi^\star}) \\
&\leq C^c \mathbb{E}_{s\sim d^\pi} \mathbb{E}_{s^\star\sim d^{\pi^\star}}[\|s - s^\star\|]
\end{aligned} \tag{3.11}$$

where we abuse the symbol $\|\cdot\|$ to denote the norm of $\mathcal{S}$ as well. The upper bound in (3.11) is an RL problem (Peng et al., 2018; Schroecker and Isbell, 2017), where the cost function namely tries to match the state distributions of the learner and the expert policies. This RL problem is perhaps easier than the original one (e.g. even if the original cost function is sparse, this new problem has a dense cost function). However, the problem in (3.11) does not have the nice supervised-learning-like property of (3.7). Therefore, we cannot reduce (3.11) to an online learning problem and run, e.g., DAGGER to update the policy. Therefore, this approach should be used as a last resort, preferred only when the action demonstrations are unavailable. We will not investigate further along this line.

3.5 Admissible Experts

Before comparing the different IL approaches, let us first think about what the desired outcome of IL looks like. Ideally we wish to design an IL algorithm that can train a policy π to perform as well as the expert $\pi^\star$ with an error that has at most *linear* dependency on the time horizon of the problem T. That is, it is desired that $J(\pi) \leq J(\pi^\star)+O(T\epsilon)$, where ϵ denotes potential errors due to approximation and optimization at each time step. Suppose otherwise an algorithm returns a policy that has performance only as $J(\pi) \leq J(\pi^\star) + O(T^2\epsilon)$; the applicability of the algorithm to tasks that involve a long problem horizon would be limited, because learning imperfection in every time step would accumulate quickly and hurt the policy performance.

It turns out this linear error dependency of $J(\pi)-J(\pi^\star)$ is not always feasible unless we make some assumptions on the quality of expert policies. Here we consider a qualification on the properties of the expert policies below.[4]

Definition 3.5.1. A policy $\pi^\star$ is called an *admissible expert* to problem (3.1) if $C^{\pi^\star} = O(1)$ independent of T.

Let us give some intuition about what Definition 3.5.1 means. For some state $s \in \mathcal{S}$ at time t, recall by definition $Q^{\pi^\star}(s, a)$ is the accumulated cost of taking some action a at time t and then executing the expert policy $\pi^\star$ afterwards. The idea behind Definition 3.5.1 is that a reasonable expert policy $\pi^\star$ should perform stably under arbitrary action perturbation, regardless of where it starts.

3.6 Comparison between Online IL and Batch IL

Now suppose that we have access to an admissible expert policy in IL. This assumption rules out the situation of learning from arbitrarily bad expert policies; in other words, we

[4]This assumption was implicitly made by Ross, Gordon, and Bagnell (2011) to derive the linear dependency bound. We note that for discrete action spaces, we just need to define the Lipschitz constant by setting the metric as the indicator function.

consider sub-optimal experts that have at least non-trivial performance with respect to (3.1). Comparing (3.6) and (3.9), we observe that in batch IL the Lipschitz constant $C^\pi(s^\star)$, without π being an admissible expert as in Definition 3.5.1, can be on the order of $T - t$ in the worst case. Therefore, if we take a uniform bound and define $C^\pi = \sup_{s \in \mathcal{S}} C^\pi(s)$, we see $C^\pi \in O(T)$. In other words, under the same assumption in online IL (i.e. (3.9) is minimized to an error in $O(T)$), the difference between $J(\pi)$ and $J(\pi^\star)$ in batch IL actually grows quadratically in T due to error compounding. This problem manifests especially in stochastic environments. In order to achieve the same level of performance as online IL, batch IL requires a more expressive policy class or more demonstration samples. As shown in (Ross, Gordon, and Bagnell, 2011), the quadratic bound is tight.

Therefore, if we have access to an admissible expert policy $\pi^\star$ that is stable in the sense of Definition 3.5.1, then online IL is preferred theoretically. This is satisfied, for example, when the expert policy is an algorithm with certain performance characteristics. However, on the contrary, when the expert is not admissible (i.e. $C^{\pi^\star} \geq \Omega(1)$), online IL would also lead to a superlinear error dependency. This could happen, e.g., when human demonstrators are adopted within online IL to perform off-road driving tasks. Because the human drivers depend heavily on instant feedback from the car to overcome stochastic disturbances, the frame-by-frame labeling approach Ross et al., 2013, for example, can lead to a very counter-intuitive, inefficient data collection process and effectively result in an inadmissible expert policy. Overall, when using human demonstrations, online IL can be as bad as batch IL (Laskey et al., 2016), simply due to inconsistencies introduced by human nature.

CHAPTER 4

IMITATION LEARNING FOR AGILE AUTONOMOUS DRIVING

4.1 Introduction

High-speed autonomous off-road driving is a challenging robotics problem (Michels, Saxena, and Ng, 2005; Williams et al., 2016, 2017) (Fig. 4.1). To succeed in this task, a robot is required to perform both precise steering and throttle maneuvers in a physically-complex, uncertain environment by executing a series of high-frequency decisions. Compared with most previously studied autonomous driving tasks, the robot here must reason about minimally-structured, stochastic natural environments and operate at high speed. Consequently, designing a control policy by following the traditional model-plan-then-act approach (Michels, Saxena, and Ng, 2005; Paden et al., 2016) becomes challenging, as it is difficult to adequately characterize the robot's interaction with the environment *a priori*.

This task has been considered previously, for example, by (Williams et al., 2016, 2017) using model-predictive control (MPC). While the authors demonstrate impressive results, their internal control scheme relies on expensive and accurate Global Positioning System (GPS) and Inertial Measurement Unit (IMU) for state estimation and demands high-frequency online replanning for generating control commands. Due to these costly hardware requirements, their robot can only operate in a rather controlled environment, which limits the applicability of their approach.

We aim to relax these requirements by designing a reflexive driving policy that uses only *low-cost, on-board* sensors (e.g. monocular camera, wheel speed sensors). Building on the success of deep reinforcement learning (RL) (Levine et al., 2016; Volodymyr et al., 2015), we adopt deep neural networks (DNNs) to parametrize the control policy and learn the desired parameters from the robot's interaction with its environment. While the use

Figure 4.1: The high-speed off-road driving task.

of DNNs as policy representations for RL is not uncommon, in contrast to most previous work that showcases RL in simulated environments (Volodymyr et al., 2015), our agent is a high-speed physical system that incurs real-world cost: collecting data is a cumbersome process, and a single poor decision can physically impair the robot and result in weeks of time lost while replacing parts and repairing the platform. Therefore, direct application of model-free RL techniques is not only sample inefficient, but costly and dangerous in our experiments.

These real-world factors motivate us to adopt *imitation learning* (IL) (Pomerleau, 1989) to optimize the control policy instead. A major benefit of using IL is that we can leverage domain knowledge through *expert* demonstrations. This is particularly convenient, for example, when there already exists an autonomous driving platform built through classic system engineering principles. While such a system (e.g. (Williams et al., 2016)) usually requires expensive sensors and dedicated computational resources, with IL we can train a lower-cost robot to behave similarly, without carrying the expert's hardware burdens over to the learner. Here we assume the expert is given as a black box oracle that can provide the desired actions when queried, as opposed to the case considered by Kahn et al. (2017) and Mordatch and Todorov (2014) where the expert can be modified to accommodate the learning progress.

In this chapter, we leverage the insights in Chapter 3 and present an IL system for real-world high-speed off-road driving tasks. By leveraging demonstrations from an algorithmic expert, our system can learn a driving policy that achieves similar performance compared

Table 4.1: Comparison of our method to prior work on IL for autonomous driving

Methods	Tasks	Observations	Action	Algorithm	Expert	Experiment
(Bojarski et al., 2016)	On-road low-speed	Single image	Steering	Batch	Human	Real &simulated
(Pomerleau, 1989)	On-road low-speed	Single image & laser	Steering	Batch	Human	Real &simulated
(Rausch et al., 2017)	On-road low-speed	Single image	Steering	Batch	Human	Simulated
(Muller et al., 2006)	Off-road low-speed	Left & right images	Steering	Batch	Human	Real
(Zhang and Cho, 2016)	On-road unknown speed	Single image	Steering + break	Online	Pre-specified policy	Simulated
(Yang et al., 2018)	On-road low/high speed	Single image + vehicle speed sequence	Steering + speed	Batch	Human	Simulated
(Yu et al., 2017)	On-road low/high speed	Image sequence + GPS/IMU measurement	Steering + acceleration	Batch	Human	Simulated
Our Method	Off-road high-speed	Single image + wheel speeds	Steering + throttle	Online	Model predictive controller	Real & simulated

to the expert. The system was implemented on a 1/5-scale autonomous AutoRally car. In real-world experiments, we show the AutoRally car—without any state estimator or online planning, but with a DNN policy that directly inputs measurements from a low-cost monocular camera and wheel speed sensors—could learn to perform high-speed driving at an average speed of $\sim$6 m/s and a top speed of $\sim$8 m/s (equivalently 108 km/h and 144 km/h on a full-scale car), matching the state-of-the-art (Williams et al., 2017). This chapter is partly based on our previous papers published as (Pan et al., 2019; Pan et al., 2018).

4.2 Related Work

End-to-end learning for self-driving cars has been explored since the late 1980s. The Autonomous Land Vehicle in a Neural Network (ALVINN) (Pomerleau, 1989) was developed to learn steering angles directly from camera and laser range measurements using a neural network with a single hidden layer. Based on similar ideas, modern self-driving cars (Bojarski et al., 2016; Muller et al., 2006; Rausch et al., 2017) have recently started to employ a batch IL approach: with DNN control policies, these systems require only expert demonstrations during the training phase and on-board measurements during the testing phase. For example, Nvidia's PilotNet (Bojarski et al., 2016, 2017), a convolutional neural network that outputs steering angle given an image, was trained to mimic human drivers' reactions to visual input with demonstrations collected in real-world road tests.

Our problem differs substantially from these previous on-road driving tasks. We study autonomous driving on a fixed set of dirt tracks, whereas on-road driving must perform well in a larger domain and contend with moving objects such as cars and pedestrians. While on-road driving in urban environments may seem more difficult, our agent must overcome challenges of a different nature. It is required to drive at high speed, on dirt tracks, the surface of which is constantly evolving and highly stochastic. As a result, high-frequency application of both steering and throttle commands are required in our task, whereas many previous work only focuses on steering commands (Bojarski et al., 2017; Muller et al.,

2006; Rausch et al., 2017). In (Yang et al., 2018), a Convolutional Neural Network (CNN) + Long Short Term Memory network (LSTM) design was proposed to predict both steering angle and vehicle speed. In (Yu et al., 2017), a convolutional LSTM model is used to predict acceleration commands. However, these approaches are not exactly end-to-end because a lower-level control module is required to compute the throttle/brake commands. Furthermore, (Yu et al., 2017) requires GPS/IMU measurements which increase the hardware cost. A Dataset Aggregation (DAGGER) (Ross, Gordon, and Bagnell, 2011) related online IL algorithm for autonomous driving was recently demonstrated in (Zhang and Cho, 2016), but only considered simulated environments and used a rule-based policy as the expert. In comparison with the previous setups, our system uses an MPC expert that solves optimal control problems at a high frequency, rather than a human driver (Bojarski et al., 2016) or a simple rule-based policy (Zhang and Cho, 2016), in order to provide timely feedback to contend with the stochasticity in high-speed dirt-track driving. A comparison of different IL approaches to autonomous driving is presented in Table 4.1.

Our task is similar to the task considered by Williams et al. (2016, 2017) and Drews et al. (2017). Compared with a DNN policy, their MPC approach has several drawbacks: computationally expensive optimization for planning is required to be performed online at high-frequency, which becomes repetitive for navigating the vehicle on a track after a few laps. In (Williams et al., 2016, 2017), accurate GPS and IMU feedbacks are also required for state estimation, which may not contain sufficient information to contend with the changing environment in off-road driving tasks. While the requirement on GPS and IMU is relaxed by using a vision-based cost map in (Drews et al., 2017), a large dataset (300,000 images) was used to train the model, expensive on-the-fly planning is still required, and speed performance is compromised. In contrast to previous work, our approach off-loads the hardware requirements to an expert. While the expert may use high-quality sensors and more computational power, our agent only needs access to cheap sensors and its control policy can run reactively in high frequency, without on-the-fly planning. Addi-

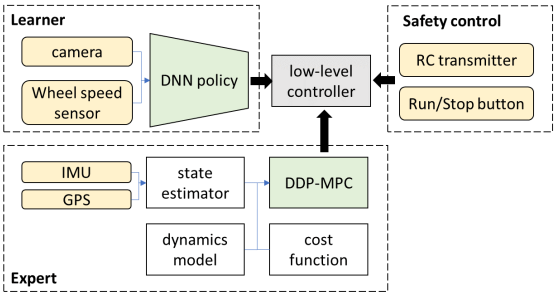


Figure 4.2: System diagram.

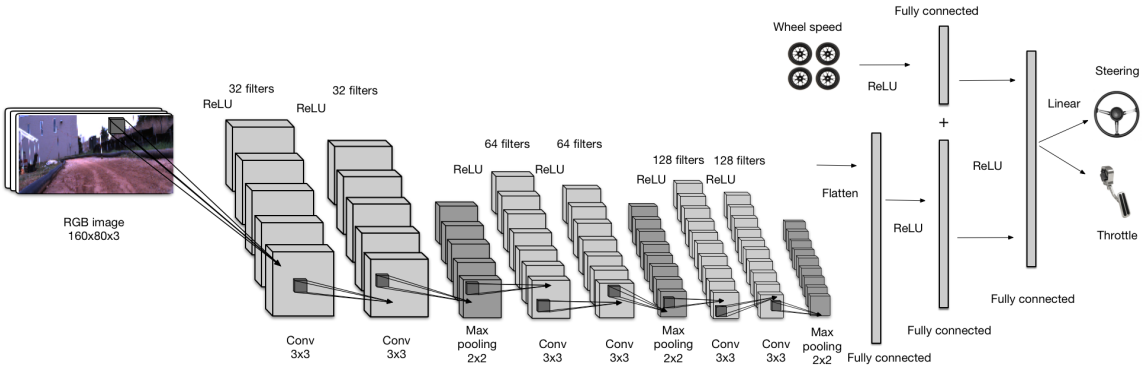


Figure 4.3: The DNN control policy.

tionally, our experimental results match those in (Williams et al., 2016), and are faster and more data efficient than that in (Drews et al., 2017).

4.3 The Autonomous Driving System

Building on the analyses in Chapter 3, we design a system that can learn to perform fast off-road autonomous driving with only on-board measurements. The overall system architecture for learning end-to-end DNN driving policies is illustrated in Fig. 4.2. It consists of three high-level controllers (an expert, a learner, and a safety control module) and a low-level controller, which receives steering and throttle commands from the high-level controllers and translates them to pulse-width modulation (PWM) signals to drive the steering and throttle actuators of a vehicle.

On the basis of the analysis in Section 3.6, we assume the expert is algorithmic and has access to expensive sensors (GPS and IMU) for accurate global state estimates[1] and

[1]Global position, heading and roll angles, linear velocities, and yaw rate.

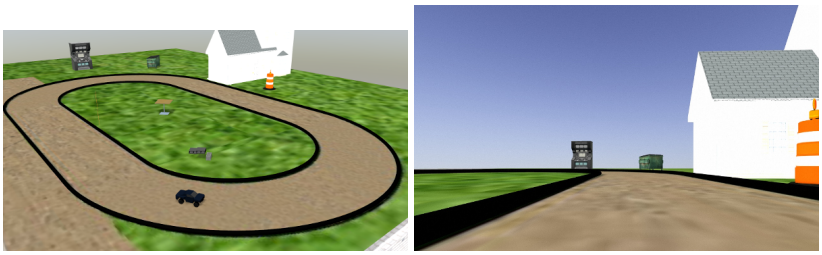


Figure 4.4: The Gazebo-based simulation enviorment (left) and a snapshot from the on-board camera (right).

resourceful computational power. The expert is built on multiple hand-engineered components, including a state estimator, a dynamics model of the vehicle, a cost function of the task, and a trajectory optimization algorithm for planning (see Section 4.3.1). By contrast, the learner is a DNN policy that has access to only a monocular camera and wheel speed sensors and is required to output steering and throttle command directly (see Section 4.3.2). In this setting, the sensors that the learner uses can be significantly cheaper than those of the expert[2]; specifically on our experimental platform, the AutoRally car (see Section 4.3.3), the IMU and the GPS sensors required by the expert in Section 4.3.1 together cost more than $6,000, while the sensors used by the learner's DNN policy cost less than $500. The safety control module has the highest priority among all three controllers and is used to prevent the vehicle from high-speed crashing.

The software system was developed based on the Robot Operating System (ROS) in Ubuntu. In addition, a Gazebo-based simulation environment (Koenig and Howard, 2004) was built (see Fig 4.4) using the same ROS interface but without the safety control module. Due to the limitation of the Gazebo simulator in terms of graphics quality and physics engine, we do not use simulated data to pre-train our control policy. The simulator was used to evaluate the performance of the software before real track tests.

[2] It might be possible to build a model that maps raw observations to state estimates, though, in general, a model may need to consider also the temporal dependency that the current state depends on the previous state. Learning such a mapping faces several challenges, however. We did not choose to do so because we wanted to completely remove the use of the expert policy, which requires real-time trajectory optimization.

4.3.1 Algorithmic Expert: Model-Predictive Control

We build the algorithmic expert based on the model-based optimal controller described in Section 4.A, which attempts to solve

$$\min_{\pi} J(\pi), \quad J(\pi) \coloneqq \mathbb{E}_{s_0,a_0,s_1,\ldots,a_{T-1}\sim\hat{\rho}^{\pi}(p)}\left[\sum_{t=0}^{T-1} c(s,a)\right], \tag{4.1}$$

in which $s \in \mathcal{S}$, $a \in \mathcal{A}$, $c(s,a)$ is the instantaneous cost, and $\hat{\rho}^{\pi}(p)$ is the distribution of trajectory generated by running the policy π with respect to the dynamics model starting from the initial state distribution p.

This algorithm is able to approximate the solution to the problem (4.1). However, in practice the task time horizon T may be very long (e.g., 1 minute), so solving the optimal problem (4.1) in a single pass is computationally inefficient and not robust to long-term prediction errors (given the fact that we never have enough data). Therefore we apply the algorithmic expert in a Model Predictive Control (MPC) fashion, which solves a shorter time horizon optimal problem at *every* sampling time: at time t, the expert policy $\pi^{\star}$ is a locally optimal policy such that

$$\pi^{\star} \approx \arg\min_{\pi} \mathbb{E}_{s_t,a_t,\ldots,a_{t+T_h}\sim\hat{\rho}^{\pi}(s_t)}\left[\sum_{\tau=t}^{t+T_h-1} c(s_\tau,a_\tau)\right] \tag{4.2}$$

where T_h is the length of horizon it previews. The details of the computation steps can be found in Section 4.A, which uses iSAM2 (Kaess et al., 2012) to estimate the vehicle states. Upon convergence, the algorithm returns a locally optimal control sequence $\{\hat{a}_t, ..., \hat{a}_{t+T_h-1}\}$, and the MPC expert executes the first action in the sequence as the expert's action at time t (i.e. $a_t^* = \hat{a}_t$). When a new vehicle state is available from the state estimator, the state-control pair is incorporated to perform incremental regression as described in Section 4.A.1. In order to ensure fast convergence, we use a 'warm-start' approach that initializes the nominal trajectory with the optimal control sequence obtained

at the previous step. Empirically the trajectory optimization algorithm converges within 3 iterations with warm-start.

In view of the analysis in Section 3.3, we can assume that the MPC expert satisfies Definition 3.5.1, because it updates the approximate solution to the model RL problem (4.1) at a high-frequency using global state information. However, because MPC requires replanning for every time step, running the expert policy (4.2) on-the-fly consumes significantly more computational power than what is required by the learner.

4.3.2 Learning a DNN Control Policy

The learner's control policy π is parametrized by a DNN containing $\sim$10 million parameters. As illustrated in Fig. 4.3, the DNN policy, consists of two sub-networks: a convolutional neural network (CNN) with 6 convolutional layers, 3 max-pooling layers, and 2 fully-connected layers, that takes 160×80 RGB monocular images as inputs,[1] and a feed-forward network with a fully-connected hidden layer that takes wheel speeds as inputs. The convolutional and max-pooling layers are used to extract lower-dimensional features from images. The DNN policy uses 3×3 filters for all convolutional layers, and rectified linear unit (ReLU) activation for all layers except the last one. Max-pooling layers with 2×2 filters are integrated to reduce the spatial size of the representation (and therefore reduce the number of parameters and computation loads). The two sub-networks are concatenated and then followed by another fully-connected hidden layer. The structure of this DNN was selected empirically based on experimental studies of several different architectures.

We consider training this DNN policy by both online IL and batch IL, which are described in Section 3.4. The online IL solves the problem

$$\min_{\pi \in \Pi} \mathbb{E}_{s,a^\star \sim \mathcal{D}} \mathbb{E}_{a \sim \pi | s} [\|a - a^\star\|], \tag{4.3}$$

[1] The raw images from the camera were re-scaled to 160×80.

and the batch IL solves the problem

$$\min_{\pi \in \Pi} \mathbb{E}_{s^\star, a^\star \sim \mathcal{D}} \mathbb{E}_{a \sim \pi | s^\star} [\|a - a^\star\|] \tag{4.4}$$

where Π denotes the effective policy class of the DNN. Comparing the two approaches, the main difference is whether the dataset $\mathcal{D}$ is collected under the learner's or the expert's distribution; in each iteration, online IL uses the newly trained learner's policy to collect more data, whereas the batch IL always the uses the expert policy to collect data.

In construction of the surrogate problem for IL, we equip the action space $\mathcal{A}$ with $\|\cdot\|_1$ for filtering outliers, and the optimization problem, (4.3) or (4.4), is solved using ADAM (Kingma and Ba, 2014), which is a stochastic gradient descent algorithm with an adaptive learning rate. Note while s or $s^\star$ is used in (4.3) or (4.4), the neural network policy does not use the state, but rather the synchronized raw observation o_t as input. Note that we did not perform any data selection or augmentation techniques in any of the experiments.[2] The only pre-processing was scaling and cropping of raw images.

4.3.3 The Autonomous Driving Platform

To validate our IL approach to off-road autonomous driving, the system was implemented on a custom-built, 1/5-scale autonomous AutoRally car (weight 22 kg; LWH 1m$\times$0.6m$\times$0.4m), shown in the top figure in Fig. 4.5. The car was equipped with an ASUS mini-ITX motherboard, an Intel quad-core i7 CPU, 16GB RAM, a Nvidia GTX 750 Ti GPU, and a 11000mAh battery. For sensors, two forward facing machine vision cameras,[3] a Hemisphere Eclipse P307 GPS module, a Lord Microstrain 3DM-GX4-25 IMU, and Hall effect wheel speed sensors were instrumented. In addition, an RC transmitter could be used to remotely control the vehicle by a human, and a physical run-stop button was installed to disable all motions in case of emergency.

[2] Data collection or augmentation techniques such as (Bojarski et al., 2016; Geist et al., 2017) can be used in conjunction with our method.

[3] In this work we only used one of the cameras.

Figure 4.5: The AutoRally car and the test track.

In the experiments, all computation was executed on-board the vehicle in real-time. In addition, an external laptop was used to communicate with the on-board computer remotely via Wi-Fi to monitor the vehicle's status. The observations were sampled and action were executed at 50 Hz to account for the high-speed of the vehicle and the stochasticity of the environment. Note this control frequency is significantly higher than (Bojarski et al., 2017) (10 Hz), (Rausch et al., 2017) (12 Hz), and (Muller et al., 2006) (15 Hz).

4.4 Experimental Setup

4.4.1 High-speed Driving Task

We tested the performance of the proposed IL system in Section 4.3 in a high-speed driving task with a desired speed of 7.5 m/s (an equivalent speed of 135 km/h on a full-scale car). The performance index of the task was formulated as the cost function in the finite-horizon RL problem with

$$c(s_t, a_t) = \alpha_1 c_{\text{pos}} + \alpha_2 c_{\text{spd}} + \alpha_3 c_{\text{slip}} + \alpha_3 c_{\text{act}}, \tag{4.5}$$

in which c_{pos} favors the vehicle to stay in the middle of the track, c_{spd} drives the vehicle to reach the desired speed, c_{slip} stabilizes the car from slipping, and c_{act} inhibits large control commands. The position cost c_{pos} for the high-speed navigation task is a 16-term cubic

function of the vehicle's global position (x, y):

$$\begin{aligned} c_{\text{pos}} &= c_0 + c_1 y + c_2 y^2 + c_3 y^3 + c_4 x + c_5 xy \\ &+ c_6 xy^2 + c_7 xy^3 + c_8 x^2 + c_9 x^2 y + c_{10} x^2 y^2 + c_{11} x^2 y^3 \\ &+ c_{12} x^3 + c_{13} x^3 y + c_{14} x^3 y^2 + c_{15} x^3 y^3. \end{aligned} \tag{4.6}$$

The coefficients $c_0, .., c_{15}$ in this cost function were identified by performing a regression to fit the track's boundary: First, a thorough GPS survey of the track was taken. Points along the inner and the outer boundaries were assigned values of -1 and $+1$, respectively, resulting in a zero-cost path along the center of the track. The coefficient values c_i were then determined by a least-squares regression of the polynomials in c_{pos} to fit the boundary data. The speed cost $c_{\text{spd}} = \|v_x - v_{\text{desired}}\|^2$ is a quadratic function which penalizes the difference between the desired speed v_{desired} and the longitudinal velocity v_x in the body frame. The side slip angle cost is defined as $c_{\text{slip}} = -\arctan(\frac{v_y}{\|v_x\|})$, where v_y is the lateral velocity in the body frame. The action cost is a quadratic function defined as $c_{\text{act}} = \gamma_1 a_1^2 + \gamma_2 a_2^2$, where a_1 and a_2 correspond to the steering and the throttle commands, respectively. In the experiments, $\gamma_1 = 1$ and $\gamma_2 = 1$ were selected.

The goal of the high-speed driving task to minimize the accumulated cost function over one-minute continuous driving. That is, under the 50-Hz sampling rate, the task horizon was set to 60 seconds ($T = 3000$). The cost information (4.5) was given to the MPC expert in Fig. 4.2 to perform online trajectory optimization with a two-second prediction horizon ($T_h = 100$). In the experiments, the weighting in (4.5) were set as $\alpha_1 = 2.5$, $\alpha_2 = 1$, $\alpha_3 = 100$ and $\alpha_4 = 60$, so that the MPC expert in Section 4.3.1 could perform reasonably well. The learner's policy was tuned by online/batch IL in attempts to match the expert's performance.

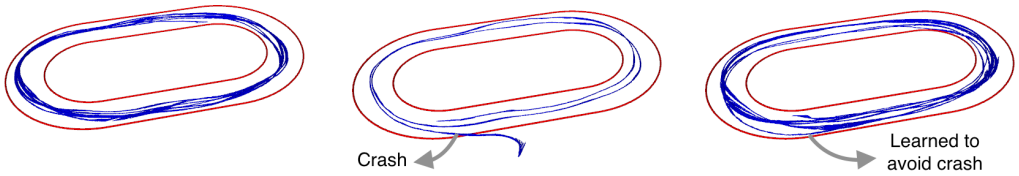
4.4.2 Test Track

All the experiments were performed on an elliptical dirt track, shown in the bottom figure of Fig. 4.5, with the AutoRally car described in Section 4.3.3. The test track was $\sim$3m wide and $\sim$30m long and built with fill dirt. Its boundaries were surrounded by soft HDPE tubes, which were detached from the ground, for safety during experimentation. Due to the changing dirt surface, debris from the track's natural surroundings, and the shifting track boundaries after car crashes, the track condition and vehicle dynamics can change from one experiment to the next, adding to the complexity of learning a robust policy.

4.4.3 Data Collection

For dynamics model learning, data was collected in two phases: first, a human driver drove the vehicle at various speeds (3 - 6 m/s) for 10 minutes, during which the state and action data were recorded, processed with spline smoothing, and re-sampled. A SSGP model of the vehicle dynamics was learned offline using the dataset (see Section 4.A.1 for details). Next, when the MPC expert (based on the learned dynamics model) was executed on the vehicle, we continued to collect new state-action data of transition dynamics and incorporated them to perform incremental updates, as described in Section 4.A.1.

For IL, training data was collected in two ways. In batch IL, the MPC expert was executed, and the camera images, wheel speed readings, and the corresponding steering and throttle commands were recorded. In online IL, a mixture of the expert and learner's policy was used to collect training data (camera images, wheel speeds, and expert actions): in the nth iteration of DAGGER, a mixed policy was executed at each time step $\hat{\pi}_n = \beta^i \pi^\star + (1 - \beta^m)\pi_{n-1}$, where π_{n-1} is learner's DNN policy after $n-1$ DAGGER iterations, and β^n is the probability of executing the expert policy. The use of a mixture policy was suggested in (Cheng and Boots, 2018; Ross, Gordon, and Bagnell, 2011) for better stability. A mixing rate $\beta = 0.6$ was used in our experiments. Note that the probability of using the expert decayed exponentially as the number of DAGGER iterations increased. Experimental



(a) MPC expert. (b) Batch IL. (c) Online IL.

Figure 4.6: Examples of vehicle trajectories, where online IL avoids the crashing case encountered by batch IL. (b) and (c) depict the test runs after training on 9,000 samples.

data was collected on an outdoor track, and consisted of changing lighting conditions and environmental dynamics. In the experiments, the rollouts about to crash were terminated remotely by overwriting the autonomous control commands with the run-stop button or the RC transmitter in the safety control module; these rollouts were excluded from the data collection.

4.4.4 Policy Learning

In online IL, three iterations of DAGGER were performed. At each iteration, the robot executed one rollout using the mixed policy described above (the probabilities of executing the expert policy were 60%, 36%, and 21%, respectively). For a fair comparison, the amount of training data collected in batch IL was the same as all of the data collected over the three iterations of online IL.

At each training phase, the optimization problem (4.3) or (4.4) was solved by ADAM for 20 epochs, with mini-batch size 64, and a learning rate of 0.001. Dropouts were applied at all fully connected layers to avoid over-fitting (with probability 0.5 for the firstly fully connected layer and 0.25 for the rest). See Section 4.3.2 for details. Finally, after the entire learning session of a policy, three rollouts were performed using the learned policy for performance evaluation.

4.5 Experimental Results

4.5.1 Algorithmic Expert vs Human Expert

First, we study the performance of the algorithmic expert. For comparison the same task was demonstrated by a human driver using a remote controller. We use speed (the faster the better) as the metric for both MPC and human driver. Other metrics such as cross-track error may not be intuitive because MPC and human driver have different objectives. For example, MPC does path following while the human driver does lane keeping, e.g., it is more desirable to not follow the center of the lane during cornering. Statistics for both MPC and human expert are shown in Table 4.2. Results were averaged over 3 independent trials. The MPC expert outperforms the human expert significantly in terms of speed (our target speed is 7.5 m/s).

While a professional race car driver could be better at utilizing the tire force potential than a hand-crafted controller (Laurense, Goh, and Gerdes, 2017), the MPC expert performs better, because, in our case, the human driver controls the vehicle via a transmitter in a third-person view, which results in a delayed response and differences in terms of sensing and handling capabilities. In addition, human drivers can be problematic for online imitation learning tasks due to the lack of instantaneous feedback from the vehicle caused by his or her own actions (as we discussed in Section 3.6 and at the beginning of Section 4.A). Labeling the actions frame-by-frame offline (Ross et al., 2013) is not possible because of the continuous throttle and steering commands. In the following experiments we focus on comparing batch and online imitation learning with the MPC expert.

4.5.2 Empirical Performance

Next we study the performance of training a control policy with online and batch IL algorithms. Fig. 4.6 illustrates the vehicle trajectories of different policies. Due to accumulating errors, the policy trained with batch IL crashed into the lower-left boundary, an area of the

Table 4.2: Test statistics. Total loss denotes the imitation loss in (3.7), which is the average of the steering and the throttle losses. Completion is defined as the ratio of the traveled time steps to the targeted time steps (3,000). All results here represent the average performance over three independent evaluation trials.

Policy	Avg. speed	Top speed	Training data	Completion ratio	Total loss	Steering/Throttle loss
Expert	6.05 m/s	8.14 m/s	N/A	100 %	0	0
Expert (human)	5.09 m/s	6.1 m/s	N/A	100 %	0	0
Batch	4.97 m/s	5.51 m/s	3000	100 %	0.108	0.092/0.124
Batch	6.02 m/s	8.18 m/s	6000	51 %	0108	0.162/0.055
Batch	5.79 m/s	7.78 m/s	9000	53 %	0.123	0.193/0.071
Batch	5.95 m/s	8.01 m/s	12000	69 %	0.105	0.125/0.083
Online (1 iter)	6.02 m/s	7.88 m/s	6000	100 %	0.090	0.112/0.067
Online (2 iter)	5.89 m/s	8.02 m/s	9000	100 %	0.075	0.095/0.055
Online (3 iter)	6.07 m/s	8.06 m/s	12000	100 %	0.064	0.073/0.055

state-action space rarely explored in the expert's demonstrations. In contrast to batch IL, online IL successfully copes with corner cases as the learned policy occasionally ventured into new areas of the state-action space.

Fig. 4.7 shows the performance in terms of distance traveled without crashing (we used the safe control module shown in Fig. 4.2 to manually terminate the rollout when the car crashed into the soft boundary) and Table 4.2 shows the statistics of the experimental results. Overall, DNN policies trained with both online and batch IL algorithms were able to achieve speeds similar to the MPC expert. However, with the same amount of training data, the policies trained with online IL in general outperformed those trained with batch IL. In particular, the policies trained using online IL achieved better performance in terms of both completion ratio and imitation loss.

In addition, we found that, when using online IL, the performance of the policy monotonically improves over iterations as data are collected, which is opposed to what was found in (Laskey et al., 2016). The discrepancy can be explained with recent theoretical analyses (Cheng and Boots, 2018; Cheng et al., 2019c; Lee et al., 2018c), which provides a necessary and sufficient condition for the convergence of the policy sequence. In particular, the authors show that adopting a non-zero mixing (as used in our experiment) is sufficient to guarantee the convergence of the learned policy sequence. Our autonomous driving system is a successful real-world demonstration of this IL theory.

Finally, it is worth noting that the traveled distance of the batch learning policy, learned with 3,000 samples, was longer than that of other batch learning policies. This is mainly because this policy achieved better steering performance than throttle performance (cf. Steering/Throttle loss in Table 4.2). That is, although the vehicle was able to navigate without crashing, it actually traveled at a much slower speed. By contrast, the other batch learning policies that used more data had better throttle performance and worse steering performance, resulting in faster speeds but also higher chances of crashing.

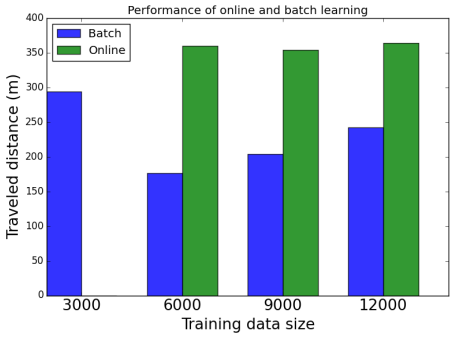


Figure 4.7: Performance of online and batch IL in the distance (meters) traveled without crashing. The policy trained with a batch of 3,000 samples was used to initialize online IL.

4.5.3 Generalizability of the Learned Policy

To further analyze the difference between the DNNs trained using online and batch IL, we embed the data in a two-dimensional space using t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008), as shown in Fig. 4.8 and Fig. 4.9. These figures visualize the data in both batch and online IL settings, where "train" denotes the data collected to train the policies and "test" denotes the data collected to evaluate the performance of the final policies after the learning phase. For the online setting, the train data include the data in all DAGGER iterations; for the batch setting, the train data include the same amount of data but collected by the expert policy. The figures plot a subset of 3,000 points from each data set.

We first observe in Fig. 4.8 that, while the wheel speed data have similar training and testing distributions, the raw image distributions are fairly misaligned. The raw images are subject to changing lighting conditions, as the policies were executed at different times and days, and to various trajectories the robot stochastically traveled. Therefore, while the task (driving fast in the same direction) is *seemingly* monotone, it actually is not. More importantly, the training and testing images were collected by executing different policies, which leads to different distributions of the neural networks inputs. This is known as the *covariate shift* problem (Shimodaira, 2000), which can significantly complicate the learning process.

The policy trained with online IL yet still demonstrated great performance in the exper-

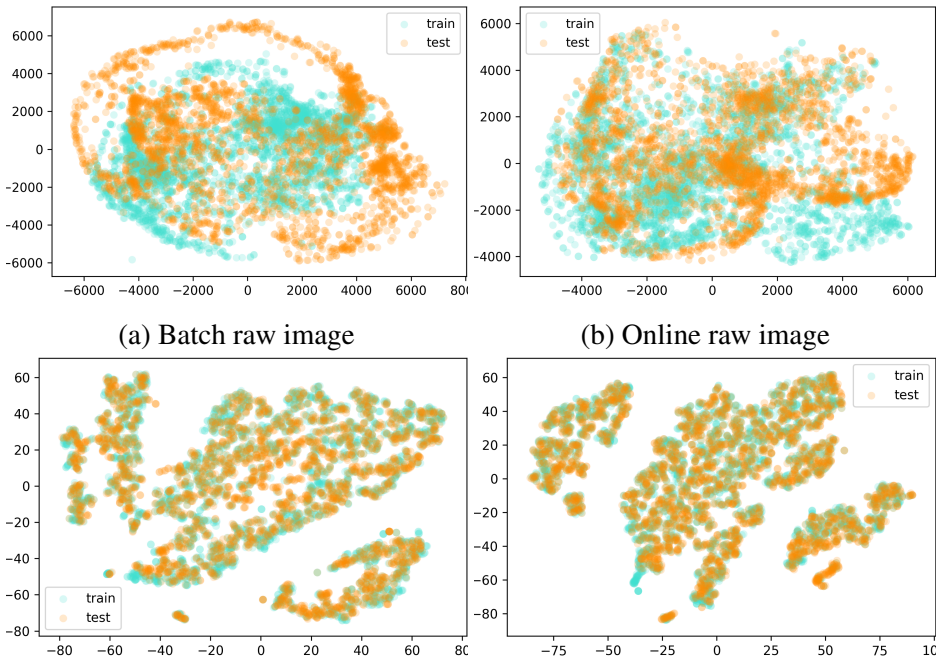
iments. To further understand how it could generalize across different image distributions, we embed its feature distribution in Fig. 4.9 (a) and (b). The feature here are the last hidden layer of the neural network; namely, the output layer is a linear function of the features. In comparison with the raw images, these abstract features (e.g. lane boundary, building shown in Fig. 4.10) extracted by the encoder CNN ideally can be more invariant across different situations.

Interestingly, despite the difference in the raw image distributions in Fig. 4.8 (a) and (b), the DNN policy trained with online IL are able to map the train and test data to similar feature distributions, as shown in Fig. 4.9 (a) and (b). An insight to this is that the online IL algorithm (e.g. DAGGER) forces the DNN to learn a set of features such that a linear combination (the last layer) of those features is sufficient to represent a good policy for a range of distributions generated during the interactive training process. In other words, the online IL paradigm effectively makes the DNN face a multi-task learning situation: it must find an invariant feature embedding that is admissible to the use of linear policies. This explains the coherency between Fig. 4.9 (a) and (b), compared with Fig. 4.8 (a) and (b).

On the contrary, the DNN policy trained with batch IL fails to learn a coherent feature embedding, as shown in Fig. 4.9 (c) and (d). (They are still better than Fig. 4.8 (a) and (b), but worse than Fig. 4.9 (a) and (b).) Based on the discussion above, this is because the DNN only needs to work well on the distribution visited by the expert policy, which is comparably simpler (an analogy to single-task learning). This could explain the inferior performance of batch IL, and its inability to deal with the corner case in Fig. 4.6 (b). This evidence shows that our online learning system can alleviate the covariate shift issue caused by executing different policies at training and testing time.

4.5.4 The Neural Network Policy

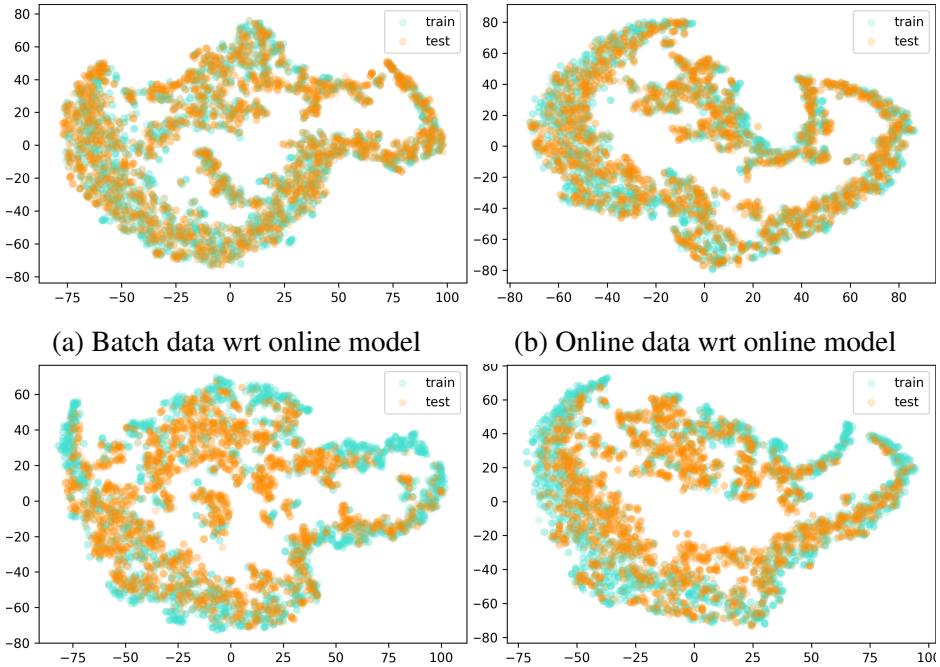
Compared with hand-crafted feature extractors, one main advantage of a DNN policy is that it can learn to extract both low-level and high-level features of an image and automatically



(a) Batch raw image (b) Online raw image

(c) Batch wheel speed (d) Online wheel speed

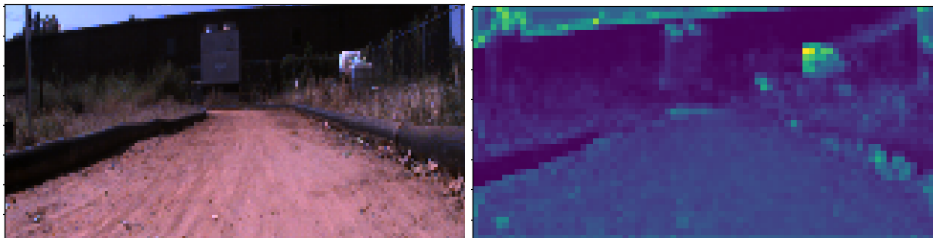
Figure 4.8: The distributions (t-SNE) of the raw images and wheel speed used as DNN policy's inputs (details in Section 4.5.3).



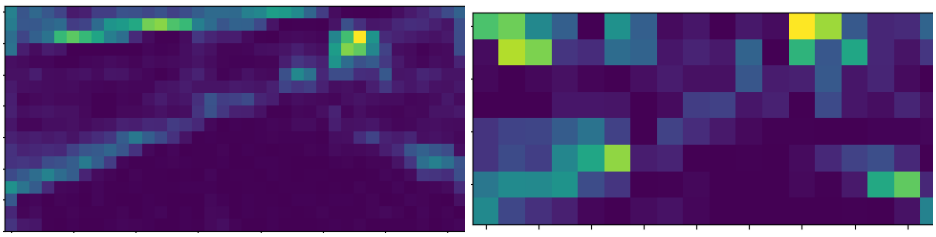
(a) Batch data wrt online model (b) Online data wrt online model

(c) Batch data wrt batch model (d) Online data wrt batch model

Figure 4.9: The distributions (t-SNE) of the learned DNN feature in the last fully-connected layer (details are in Section 4.5.3).



(a) raw image (b) max-pooling1



(c) max-pooling2 (d) max-pooling3

Figure 4.10: The input RGB image and the averaged feature maps for each max-pooling layer.

detect the parts that have greater influence on steering and throttle. We validate this idea by showing in Fig. 4.10 the averaged feature map at each max-pooling layer (see Fig. 4.3), where each pixel represents the averaged unit activation across different filter outputs. We can observe that at a deeper level, the detected salient features are boundaries of the track and parts of a building. In contrast, grass and dirt contribute little.

We also analyze the importance of incorporating wheel speeds in our task. We compare the performance of the policy based on our DNN policy and a policy based on only the CNN subnetwork (without wheel-speed inputs) in batch IL. The data was collected in accordance with Section 4.4.3. Fig. 4.11 shows the batch IL loss in (4.4) of different network architectures. The full DNN policy in Fig. 4.3 achieved better performance consistently. While images contain position and orientation information, it is insufficient to infer velocities, which are a part of the (hidden) vehicle state. Therefore, we conjecture state-of-the-art CNNs (e.g. (Bojarski et al., 2017)) cannot be directly used to perform both lateral and longitudinal controls, as we do here. By contrast, while without a recurrent architecture, our DNN policy learned to combine wheel speeds in conjunction with CNN to infer hid-

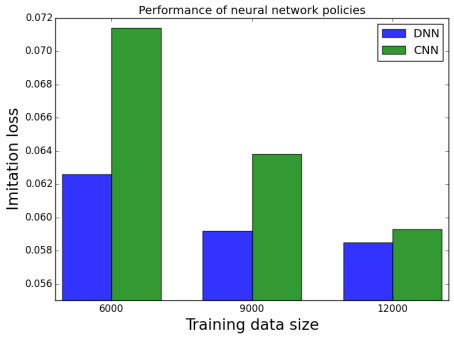


Figure 4.11: Performance comparison between our DNN policy and its CNN sub-network in terms of batch IL loss, where the horizontal axis is the size of data used to train the neural network policies.

den state and achieve better performance. Recent work has shown that recurrent neural networks (such as LSTM) can be used to predict speed (Yang et al., 2018) or acceleration commands (Yu et al., 2017). However, a lower-level control module is required to compute the throttle commands, therefore the learned policy is not end-to-end. Incorporating recurrent structures into our imitation learning framework could be a interesting extension of this work and is left to future research.

4.6 Conclusion

We introduce an end-to-end system to learn a deep neural network control policy for high-speed driving that maps raw on-board observations to steering and throttle commands by mimicking a model predictive controller. In real-world experiments, our system was able to perform fast off-road navigation autonomously using a low-cost monocular camera and wheel speed sensors. We also provide an analysis of both online and batch IL frameworks, both theoretically and empirically and show that our system, when trained with online IL, learns generalizable features that are more robust to covariate shift than features learned with batch IL.

4.A Design of Algorithmic Expert

In Chapter 3, we showed that if an admissible expert is available then online IL (e.g. DAGGER) provides a learning framework that can achieve the desirable linear error dependency. Due to the dynamic nature of our high-speed driving task, we consider *algorithmic* experts, because human experts might not be able to provide stable and consistent high-frequency feedbacks while the learner is driving the car.

More precisely, we recall that online IL requires action demonstrations on the trajectories generated by running the learner's policy. For high-speed driving, this means that human experts need to provide action demonstrations (desired steering and throttle commands here) when the vehicle is being autonomously controlled by the (suboptimal) learner's policy. Deprived of the usual sensory-motor feedback, human drivers often provide poor feedback: for example, we have observed that human drivers tend to overcompensate when providing steering when faced with unexpected vehicle dynamics (under the control of the learner's policy). This inconsistency can introduce bias into the demonstrated actions, in the worst case, effectively creating an inadmissble expert policy for IL (see Section 3.6).

By contrast, a natural candidate with such stability property would be the optimal policy of (3.1). Specifically, suppose the dynamics of (3.1) is known, an expert policy can be obtained by solving problem (3.1) via Dynamic Programming, and its value function is the solution to the Bellman equation

$$V_t^{\pi\star}(s_t) = \min_{a_t \in \mathcal{A}} c(s_t, a_t) + \mathbb{E}_{s_{t+1}\sim\mathcal{P}|s_t,a_t}\left[V_{t+1}^{\pi\star}(s_{t+1})\right] \tag{4.7}$$

where $\mathcal{P}(s'|s,a)$ denotes the distribution of vehicle dynamics (i.e. the state transition). (We made the dependency on time explicit here.)

However, in practice, the above idealistic approach faces two main challenges: 1) the transition probability $\mathcal{P}(s'|s,a)$ is hard to obtain due to the complexity of vehicle dynamics at high-speed in off-road conditions. 2) Solving (4.7) for all $s \in \mathcal{S}$ is computationally

intractable due to the curse of dimensionality of Dynamic Programming. In this work, we address these two challenges using a probabilistic dynamics model and trajectory optimization. We describe these techniques in the following and they will be used as the foundation to design the algorithmic expert in our IL system, as later described in Section 4.3.1.

4.A.1 Probabilistic Dynamics Model

Under normal driving conditions, a planar single-track vehicle model derived from Newtonian physics (Kong et al., 2015) and an empirical tire model (Rajamani, 2011) are widely used and usually sufficient for control design. In contrast, controlling a race car in aggressive maneuvers, e.g., cornering at the limit of tire-road friction, requires more sophisticated techniques to estimate the tire-road friction coefficient (Laurense, Goh, and Gerdes, 2017). In our case, the friction changes rapidly due to the uneven dirt surface, which makes it more challenging to estimate the coefficient. In practice, physics-based models do not capture the aforementioned dynamics effects well, and neural networks (NNs) have been used for vehicle dynamics model identification (Rutherford and Cole, 2010; Williams et al., 2017). However, NNs typically do not adapt to rapidly changing road conditions in real-time. In addition, NNs do not provide estimates of model uncertainty given limited amount of training data, which can hamper accurate long-range prediction. Motivated by these challenges, we consider learning a probabilistic model, Sparse Spectrum Gaussian Processes (Lázaro-Gredilla et al., 2010) (SSGPs) from data to approximate the vehicle dynamics, and Bayesian inference to predict the vehicle's future states.

SSGP Regression

Gaussian process regression (GPR) (Williams and Rasmussen, 2006) is a principled way to perform Bayesian inference in function space. Consider the task of learning function $f : \mathbb{R}^d \to \mathbb{R}$ (the vehicle dynamics model in our case, and we treat each output dimension

independently) given a dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ that are sampled according to

$$y_n = f(x_n) + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \sigma^2), \tag{4.8}$$

where ϵ is an independent additive zero-mean Gaussian noise with covariance σ^2. Data collection details will be provided in Section 4.4.3. GPR reasons about potential candidates of the latent function, under the assumption that f has a prior GP distribution $f \sim \mathcal{GP}(\bar{m}, k)$, with mean function $\bar{m} : \mathbb{R}^d \to \mathbb{R}$ and covariance function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. That is, for any $x, x' \in \mathbb{R}^d$, $\mathbb{E}[f(x)] = \bar{m}(x)$, $\mathbb{C}[f(x), f(x')] = k(x, x')$, and for any finite subset $\{x_n \in \mathbb{R}^d\}_{n=1}^K$, $\{f(x_n)\}_{n=1}^K$ is Gaussian distributed. Using the dataset $\mathcal{D}$, the inference problem of GRP computes the posterior distribution of the latent function f. Without loss of generality, we assume $\bar{m}(x) = 0$ *a priori*.

While theoretically sound, the exact inference problem of GPR is challenging for large datasets due to its $O(N^3)$ time and $O(N^2)$ space complexities (Williams and Rasmussen, 2006), which is a direct consequence of storing and inverting an $N \times N$ Gram matrix. Many approximate techniques have been proposed to tackle this challenge, including using random Fourier features (Lázaro-Gredilla et al., 2010), degenerate priors (Snelson and Ghahramani, 2006), variational posteriors (Cheng and Boots, 2017; Hensman, Fusi, and Lawrence, 2013; Salimbeni et al., 2018; Titsias, 2009). In this work, we adopt the approach by Lázaro-Gredilla et al. (2010) for its implementation simplicity.

To reduce the complexity, Lázaro-Gredilla et al. (2010) use approximate GP models, SSGPs, which are a class of GPs with prior covariance function in the form:

$$k(x, x') = \phi(x)^\top \phi(x') + \sigma^2 \delta(x - x'), \ \phi(x) = \begin{bmatrix} \phi^c(x) \\ \phi^s(x) \end{bmatrix},$$
$$\phi^c(x) = \begin{bmatrix} \phi_1^c(x) & \dots & \phi_m^c(x) \end{bmatrix}^\top, \phi^s(x) = \begin{bmatrix} \phi_1^s(x) & \dots & \phi_m^s(x) \end{bmatrix}^\top,$$
$$\phi_i^c(x) = \eta \cos(\omega_i^\top x), \ \phi_i^s(x) = \eta \sin(\omega_i^\top x), \ \omega_i \sim p(\omega),$$

where function $\phi : \mathbb{R}^d \to \mathbb{R}^{2m}$ is an explicit finite-dimensional feature map [4], η is a scalar scaling coefficient, δ is the Kronecker delta function, and vector ω_i is sampled according to some spectral density $p(\omega)$ function. Based on Bochner's theorem, it can be shown that SSGPs can unbiasedly approximate any continuous shift-invariant kernels if $p(\omega)$ is constructed properly with respect to the original covariance function (Lázaro-Gredilla et al., 2010).

Because of the explicit finite-dimensional feature map ϕ, each SSGP is equivalent to a Gaussian distribution over the weights of the features $w \in \mathbb{R}^{2m}$ and has a prior distribution of weights w as $\mathcal{N}(0, I)$ (Lázaro-Gredilla et al., 2010). Given a fixed feature map, the posterior distribution of w conditioned on the data $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$ is

$$w \sim \mathcal{N}(\alpha,\ \sigma^2 A^{-1}), \tag{4.9}$$

$$\alpha = A^{-1}\Phi Y, \quad A = \Phi\Phi^\top + \sigma^2 I, \tag{4.10}$$

which can be derived through Bayesian linear regression. In (4.10), the column vector Y and the matrix Φ are specified by the data $\mathcal{D}$, in which $Y = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}^\top$ and $\Phi = \begin{bmatrix} \phi(x_1) & \dots & \phi(x_n) \end{bmatrix}$. Consequently, the posterior distribution over the output y in (4.8) at a test point x is *exactly Gaussian*

$$p(y|x) = \mathcal{N}(\alpha^\top \phi(x),\ \sigma^2 + \sigma^2 \|\phi(x)\|^2_{A^{-1}}). \tag{4.11}$$

in which the posterior variance explicitly captures the model uncertainty in predicting $f(x)$.

We use this SSGP framework to model the state transition of the unknown vehicle dynamics (i.e. the latent function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ that determines the change of state $\Delta s_t \coloneqq s_{t+1} - s_t$ given a concatenated state-action pair (s_t, a_t) as input). We assume each output dimension is conditionally independent[5] and use a SSGP model for each out-

[4] ϕ is obtained by concatenating m random features into a vector form.

[5] We assume that, for outputs in different dimensions y_a and y_b, $p(y_a, y_b|x) = p(y_a|x)p(y_b|x)$.

put dimension. The hyper-parameters σ, η are optimized via maximizing the GP marginal likelihood (Williams and Rasmussen, 2006).

Incremental Update

In order to cope with rapidly changing dynamics (e.g, caused by stochastic road conditions), when a new vehicle state is available, we incorporate it to incrementally *update* the posterior distribution over weightz w in (4.9) of the SSGP dynamics model. We note this can be done rather efficiently without storing and inverting the A matrix explicitly. Instead we keep track of its Cholesky factor R where $A = R^\top R$ and perform rank-1 update given a new sample (Gijsberts and Metta, 2013). The computation requires $O(m^2)$ time and can be performed in real-time if m is moderate. To cope with time-varying dynamics, we employ a forgetting factor $\lambda \in (0, 1)$ such that the previous samples' impact decays exponentially in time (Ljung, 1998). Details are omitted.

Multi-step Prediction

We need to be able to perform multi-step prediction in order to use the SSGP dynamics model inside a trajectory optimization algorithm. We provide the details of information propagation across SSGP dynamics models in the following. At time t, suppose the distribution at the current state s_t is distributed according to $p(s_t) = \mathcal{N}(\mu_t, \Sigma_t)$ and the current action is a_t. We wish to compute the state distribution $p(s_{t+1})$ at time $t+1$, which is related to the current state s_t through

$$s_{t+1} = s_t + f(s_t, a_t) + w_t, \quad w_t \sim \mathcal{N}(0, \sigma^2), \tag{4.12}$$

where f is a random function given by the SSGP model. As in general $p(s_{t+1})$ can be quite complicated, in this work, we approximate the predictive distribution with a Gaussian distribution $p(s_{t+1}) \approx \mathcal{N}(\mu_{t+1}, \Sigma_{t+1})$ through linearizing the predictive mean function of

the SSGP model. The moments of this approximate Gaussian predictive distribution can be represented as follows (Pan et al., 2017b):

$$\mu_{t+1} = \mu_t + \mathbb{E}[\Delta s_t] \tag{4.13}$$
$$\Sigma_{t+1} = \Sigma_t + \mathrm{Var}[\Delta s_t] + \mathrm{Cov}(s_t, \Delta s_t) + \mathrm{Cov}(\Delta s_t, s_t).$$

Equivalently, we can write the propagation of statistics in (4.13) in terms of the belief of state s_t. Define the belief as $b_t = [\mu_t \ \mathrm{vec}(\Sigma_t)]^\top$, where $\mathrm{vec}(\Sigma_t)$ is the vectorization of Σ_t, and denote the space of all beliefs by $\mathbb{B} \subset \mathbb{R}$. We can write (4.13) in a compact form as

$$b_{t+1} = \mathbf{F}(b_t, a_t), \tag{4.14}$$

where $\mathbf{F} : \mathbb{B} \times \mathcal{A} \to \mathbb{B}$ is the effective map by (4.13). This new equation corresponds to the belief-space representation of the dynamics; below we introduce a trajectory optimization method to obtain optimal actions based on the belief-space dynamics model in (4.14).

4.A.2 Trajectory Optimization

Solving (4.7) globally is notoriously difficult, requiring discretization of the state space $\mathcal{S}$ and incurs exponentially large complexity. In order to solve the optimal control problem efficiently, we use a trajectory optimization method, Differential Dynamic Programming (DDP) (Jacobson and Mayne, 1970), which approximates the solution to (4.7) locally around a trajectory. To control the vehicle under model uncertainty, we use the belief-space dynamics model in (4.14). The instantaneous cost function $l : \mathbb{B} \times \mathcal{A} \to \mathbb{R}$ defined as $l(b, a) = \mathbb{E}_s[c(s, a)|b]$ where the cost $c(s, a)$ is designed to 1) keep the car close to the middle of the track, 2) travel at a target speed, 3) stabilize the car from slipping, and 4) minimize throttle, brake and steering efforts. The details will be described in Section 4.4.1.

DDP is an iterative method. At each iteration, it creates a local model along a nominal trajectory in the belief space including: 1) a linear approximation of the dynamics model; 2)

a second-order approximation of the value function. Denote the belief and control nominal trajectory as $(\bar{b}_{1:T}, \bar{a}_{1:T})$ and deviations from this trajectory as $\delta b_t = b_t - \bar{b}_t$, $\delta a_t = a_t - \bar{a}_t$. The linear approximation of the belief dynamics along the nominal trajectory is

$$\delta b_{t+1} = \mathbf{F}_t^b \delta b_t + \mathbf{F}_t^a \delta a_t. \tag{4.15}$$

where $\mathbf{F}_t^b, \mathbf{F}_t^a$ are Jacobian matrices and the superscripts denote the variables involved in the partial derivatives[6]. Given the closed-form expression of $\mathbf{F}$, these derivatives can be evaluated efficiently without using numerical differentiation techniques. To propagate the value function, we construct a quadratic approximation of the instantaneous cost function l along the nominal belief and control trajectory, i.e.,

$$l(b_t, a_t) \approx l_t^0 + (l_t^b)^\top \delta b_t + (l_t^a)^\top \delta a_t + \frac{1}{2} \begin{bmatrix} \delta b_t \\ \delta a_t \end{bmatrix}^\top \begin{bmatrix} l_t^{bb} & l_t^{bu} \\ l_t^{ub} & l_t^{uu} \end{bmatrix} \begin{bmatrix} \delta b_t \\ \delta a_t \end{bmatrix}, \tag{4.16}$$

where $l_t^0 = l(\bar{b}_t, \bar{a}_t)$. Given the above local approximations of dynamics (4.15) and cost function (4.16), DDP creates a quadratic model of the value function

$$V_t^{\pi^\star}(b) = \min_{a_t \in \mathcal{A}} \big(l(b_t, a_t) + V_t^{\pi^\star}\big(\mathbf{F}(b_t, a_t)\big)\big) \tag{4.17}$$

$$\approx V_t^0 + (V_t^b)^\top \delta b_t + \frac{1}{2} \delta b_t^\top V_t^{bb} \delta b_t. \tag{4.18}$$

In order to compute V_t^0, V_t^b, V_t^{bb}, we define the term inside the $\min$ operator in (4.17) as the Q-function

$$Q_t^{\pi^\star}(b_t, a_t) = l(b_t, a_t) + V_t^{\pi^\star}\big(\mathbf{F}(b_t, a_t)\big).$$

[6]We will use this superscript rule for dynamics and cost-related terms.

Then we expand it up to the second order in δb and δa along $\bar{b}_t, \bar{a}_t$.

$$Q_t^{\pi^\star}(\bar{b}_t + \delta b_t, \bar{a}_t + \delta a_t) \approx Q_t^0 + Q_t^b \delta b_t + Q_t^a \delta a_t + \frac{1}{2} \begin{bmatrix} \delta b_t \\ \delta a_t \end{bmatrix}^\top \begin{bmatrix} Q_t^{bb} & Q_t^{ba} \\ Q_t^{ab} & Q_t^{aa} \end{bmatrix} \begin{bmatrix} \delta b_t \\ \delta a_t \end{bmatrix}, \tag{4.19}$$

and find an optimized control law by minimizing (4.19), i.e.,

$$\begin{aligned} \delta \hat{a}_t &= \arg\min_{\delta a_t} \left[Q(\bar{b}_t + \delta b_t, \bar{a}_t + \delta a_t) \right] \\ &= -(Q_t^{aa})^{-1}(Q_t^a + Q_t^{ab} \delta b_t). \end{aligned} \tag{4.20}$$

The quadratic model (4.17) of the value function V_t^0, V_t^b, V_t^{bb} can be computed in a backward pass by inserting the optimized control law $\hat{a}_t = \bar{a}_t + \delta \hat{a}_t$ into (4.19), see (Jacobson and Mayne, 1970; Tassa, Erez, and Smart, 2008) for details. Once the optimized control law along the entire nominal trajectory is computed through the backward pass, it is applied to the dynamics (4.14) to generate a new nominal trajectory in a forward pass. This backward-forward scheme is repeated for multiple iterations until convergence.

Unlike Quadratic programming (QP)-based approaches (Borrelli et al., 2005), our DDP-based approach is self-contained and does not rely on an external optimization solver. Compared to sampling-based method (Wagener et al., 2019; Williams et al., 2017) that uses massive forward simulations, our approach is more efficient as it exploits of the structure of the dynamics model (4.14).

CHAPTER 5

FAST POLICY LEARNING THROUGH IMITATION AND REINFORCEMENT

5.1 Introduction

In Chapter 4, we show that imitation learning (IL), which works by leveraging additional information provided through expert demonstrations, can be used as an alternate strategy to reinforcement learning (RL) for faster policy learning (Pomerleau, 1989; Schaal, 1999). However, despite significant recent breakthroughs in our understanding of imitation learning (Cheng and Boots, 2018; Ross, Gordon, and Bagnell, 2011), the performance of IL is still highly dependent on the quality of the expert policy. When only a suboptimal expert is available, policies learned with standard IL can be inferior to the policies learned by tackling the RL problem directly with approaches such as policy gradients.

Several recent attempts have endeavored to combine RL and IL (Chang et al., 2015; Nair et al., 2017; Rajeswaran et al., 2017; Ross and Bagnell, 2014; Sun, Bagnell, and Boots, 2018). These approaches incorporate the cost information of the RL problem into the imitation process, so the learned policy can *both* improve faster than their RL-counterparts and outperform the suboptimal expert policy. Despite reports of improved empirical performance, the theoretical understanding of these combined algorithms are still fairly limited (Rajeswaran et al., 2017; Sun, Bagnell, and Boots, 2018). Furthermore, some of these algorithms have requirements that can be difficult to satisfy in practice, such as state resetting (Chang et al., 2015; Ross and Bagnell, 2014).

In this chapter, we aim to provide an algorithm that combines the best aspects of RL and IL. We accomplish this by first formulating first-order RL and IL algorithms in a common mirror descent framework, and show that these algorithms can be viewed as a single approach that only differs in the choice of first-order oracle. On the basis of this new

insight, we address the difficulty of combining IL and RL with a simple, randomized algorithm, named LOKI (Locally Optimal search after K-step Imitation). As its name suggests, LOKI operates in two phases: picking K randomly, it first performs K steps of online IL and then improves the policy with a policy gradient method afterwards. Compared with previous methods that aim to combine RL and IL, LOKI is extremely straightforward to implement. Furthermore, it has stronger theoretical guarantees: by properly randomizing K, LOKI performs as if directly running policy gradient steps with the expert policy as the initial condition. Thus, not only can LOKI improve faster than common RL methods, but it can also significantly outperform a suboptimal expert. This is in contrast to previous methods, such as AGGREVATTE (Ross and Bagnell, 2014), which generally cannot learn a policy that is better than a one-step improvement over the expert policy. In addition to these theoretical contributions, we validate the performance of LOKI in multiple simulated environments. The empirical results corroborate our theoretical findings. This chapter is partly based on our paper published as (Cheng et al., 2018a).

5.2 Problem Setup

In this chapter, we consider solving discrete-time γ-discounted infinite-horizon RL problems with $\gamma \in [0, 1)$.[1] Let $\mathcal{S}$ and $\mathcal{A}$ be the state and the action spaces, and let Π be the policy class. The objective is to find a policy $\pi \in \Pi$ that minimizes an accumulated cost $J(\pi)$ defined as

$$\min_{\pi \in \Pi} J(\pi), \quad J(\pi) \coloneqq \mathbb{E}_{s_0, a_0, \ldots \sim \rho_\pi(p)} \left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \right], \tag{5.1}$$

in which $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$, c is the instantaneous cost, and $\rho^\pi(p)$ denotes the distribution of trajectories $(s_0, a_0, s_1, \ldots)$ generated by running the stationary policy π starting from a fixed initial state distribution $p(s_0)$.

[1] LOKI can be easily adapted to finite-horizon problems.

We denote $Q^\pi(s,a)$ as the Q-function under policy π and $V^\pi(s) = \mathbb{E}_{a\sim\pi|s}[Q^\pi(s,a)]$ as the associated value function, where $\pi(a|s)$ denotes the action distribution given state $s \in \mathcal{S}$. In addition, we denote $d_t^\pi(s)$ as the state distribution at time t generated by running the policy π for the first t steps, and we define the average state distribution $d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_t^\pi(s)$. Note that, while we use the notation $\mathbb{E}_{a\sim\pi|s}$, the policy class Π can be either deterministic or stochastic.

We generally will not deal with the objective function in (5.1) directly. Instead, we consider a surrogate problem

$$\min_{\pi\in\Pi} \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}[A^{\pi'}(s,a)], \tag{5.2}$$

where $A^{\pi'} = Q^{\pi'} - V^{\pi'}$ is the (dis)advantage function with respect to some fixed reference policy π'. For compactness of writing, we will often omit the random variable in expectation; e.g., the objective function in (5.2) will be written as $\mathbb{E}_{d^\pi}\mathbb{E}_\pi[A^{\pi'}]$ for the remainder of paper. By the performance difference lemma below (Kakade and Langford, 2002), it is easy to see that solving (5.2) is equivalent to solving (5.1). (See also Chapter 2.)

Lemma 5.2.1. *(Kakade and Langford, 2002) Let π and π' be two policies and $A^{\pi'}(s,a) = Q^{\pi'}(s,a) - V^{\pi'}(s)$ be the (dis)advantage function with respect to running π'. It holds that*

$$J(\pi) = J(\pi') + \frac{1}{1-\gamma}\mathbb{E}_{d^\pi}\mathbb{E}_\pi[A^{\pi'}]. \tag{5.3}$$

5.3 First-Order RL and IL

We formulate both first-order RL and IL methods within a single mirror descent framework (Nemirovski et al., 2009), which includes common update rules (Kakade, 2002; Peters, Mülling, and Altun, 2010; Peters and Schaal, 2008; Rawlik, Toussaint, and Vijayakumar, 2012; Ross, Gordon, and Bagnell, 2011; Schulman et al., 2015a; Silver et al., 2014; Sun et al., 2017; Sutton et al., 2000). We show that policy updates based on RL and IL

mainly differ in first-order stochastic oracles used, as summarized in Table 5.1.

Table 5.1: Comparison of First-Order Oracles

Method	First-Order Oracle
POLICY GRADIENT (Section 5.3.2)	$\mathbb{E}_{d^{\pi_n}}\left(\nabla_\theta \mathbb{E}_\pi\right)\left[A^{\pi_n}\right]$
DAGGERED (Section 5.3.2)	$\mathbb{E}_{d^{\pi_n}}\left(\nabla_\theta \mathbb{E}_\pi\right)\left[\mathbb{E}_{\pi^\star}[m]\right]$
AGGREVATED (Section 5.3.2)	$\mathbb{E}_{d^{\pi_n}}\left(\nabla_\theta \mathbb{E}_\pi\right)\left[A^{\pi^\star}\right]$
SLOLS (Section 5.6)	$\mathbb{E}_{d^{\pi_n}}\left(\nabla_\theta \mathbb{E}_\pi\right)\left[(1-\lambda)A^{\pi_n} + \lambda A^{\pi^\star}\right]$
THOR (Section 5.6)	$\mathbb{E}_{d^{\pi_n}}\left(\nabla_\theta \mathbb{E}_\pi\right)\left[A^{H,\pi^\star}_{\pi_n,t}\right]$

5.3.1 Mirror Descent

We begin by defining the iterative rule to update policies. We assume that the learner's policy π is parametrized by some $\theta \in \Theta$, where Θ is a closed and convex set, and that the learner has access to a family of strictly convex functions $\mathcal{R}$.

To update the policy, in the nth iteration, the learner receives a vector g_n from a first-order oracle, picks $R_n \in \mathcal{R}$, and then performs a mirror descent step:

$$\theta_{n+1} = P_{n,g_n}(\theta_n) \tag{5.4}$$

where P_{n,g_n} is a prox-map defined as

$$P_{n,g_n}(\theta_n) = \arg\min_{\theta\in\Theta} \langle g_n, \theta\rangle + \frac{1}{\eta_n} B_{R_n}(\theta||\theta_n). \tag{5.5}$$

$\eta_n > 0$ is the step size, and B_{R_n} is the Bregman divergence associated with R_n (Bregman, 1967): $B_{R_n}(\theta||\theta_n) \coloneqq R_n(\theta) - R_n(\theta_n) - \langle \nabla R_n(\theta_n), \theta - \theta_n\rangle$.

By choosing proper R_n, the mirror descent framework in (5.4) covers most RL and IL algorithms. Common choices of R_n include negative entropy (Peters, Mülling, and Altun, 2010; Rawlik, Toussaint, and Vijayakumar, 2012), $\frac{1}{2}\|\theta\|_2^2$ (Silver et al., 2014; Sutton et al.,

2000), and $\frac{1}{2}\theta^\top F(\theta_n)\theta$ with $F(\theta_n)$ as the Fisher information matrix (Kakade, 2002; Peters and Schaal, 2008; Schulman et al., 2015b).

5.3.2 First-Order Oracles

While both first-order RL and IL methods can be viewed as performing mirror descent, they differ in the choice of the first-order oracle that returns the update direction g_n. Here we show the vector g_n of both approaches can be derived as a stochastic approximation of the (partial) derivative of $\mathbb{E}_{d^\pi}\mathbb{E}_\pi[A^{\pi'}]$ with respect to policy π, but with a different reference policy π'.

Policy Gradients

A standard approach to RL is to treat (5.1) as a stochastic nonconvex optimization problem. In this case, g_n in mirror descent (5.4) is an estimate of the policy gradient $\nabla_\theta J(\pi)$ (Sutton et al., 2000; Williams, 1992).

To compute the policy gradient in the nth iteration, we set the current policy π_n as the reference policy in (5.3) (i.e. $\pi' = \pi_n$), which is treated as constant in θ in the following policy gradient computation. Because $\mathbb{E}_{\pi_n}[A^{\pi_n}] = \mathbb{E}_{\pi_n}[Q^{\pi_n}] - V^{\pi_n} = 0$, using (5.3), the policy gradient can be written as[2]

$$\begin{aligned}(1-\gamma)\nabla_\theta J(\pi)|_{\pi=\pi_n} &= \nabla_\theta \mathbb{E}_{d^\pi}\mathbb{E}_\pi[A^{\pi_n}]|_{\pi=\pi_n} \\ &= \left(\nabla_\theta \mathbb{E}_{d^\pi}\right)[0] + \mathbb{E}_{d^\pi}\left(\nabla_\theta \mathbb{E}_\pi\right)[A^{\pi_n}]|_{\pi=\pi_n} \\ &= \mathbb{E}_{d^\pi}\left(\nabla_\theta \mathbb{E}_\pi\right)[A^{\pi_n}]|_{\pi=\pi_n} \end{aligned} \tag{5.6}$$

The above expression is unique up to a change of baselines: $\left(\nabla_\theta \mathbb{E}_\pi\right)[A^{\pi_n}]$ is equivalent to $\left(\nabla_\theta \mathbb{E}_\pi\right)[A^{\pi_n} + b]$, because $\left(\nabla_\theta \mathbb{E}_\pi\right)[b(s)] = \nabla_\theta b(s) = 0$, where $b : \mathcal{S} \to \mathbb{R}$ is also called a control variate (Greensmith, Bartlett, and Baxter, 2004).

[2]We assume the cost is sufficiently regular so that the order of differentiation and expectation can exchange.

The exact formulation of $(\nabla_\theta \mathbb{E}_\pi)\left[A^{\pi_n}\right]$ depends on whether the policy π is stochastic or deterministic. For stochastic policies,[3] we can compute it with the likelihood-ratio method and write

$$(\nabla_\theta \mathbb{E}_\pi)\left[A^{\pi_n}\right] = \mathbb{E}_\pi[A^{\pi_n} \nabla_\theta \log \pi] \tag{5.7}$$

For deterministic policies, we replace the expectation as evaluation (as it is the expectation over a Dirac delta function, i.e. $a = \pi(s)$) and use the chain rule:

$$(\nabla_\theta \mathbb{E}_\pi)\left[A^{\pi_n}\right] = \nabla_\theta A^{\pi_n}(s, \pi) = \nabla_\theta \pi \nabla_a A^{\pi_n} \tag{5.8}$$

Substituting (5.7) or (5.8) back into (5.6), we get the equation for stochastic policy gradient (Sutton et al., 2000) or deterministic policy gradient (Silver et al., 2014). Note that the above equations require the exact knowledge, or an unbiased estimate, of A^π. In practice, these terms are further approximated using function approximators, leading to biased gradient estimators (Konda and Tsitsiklis, 2000; Mnih et al., 2016; Schulman et al., 2015a).

Imitation Gradients

An alternate strategy to RL is IL. In particular, we consider *online* IL, which interleaves data collection and policy updates to overcome the covariate shift problem of traditional batch IL (Ross, Gordon, and Bagnell, 2011). Online IL assumes that a (possibly suboptimal) expert policy $\pi^\star$ is available as a black-box oracle, from which demonstrations $a^\star \sim \pi^\star(a^\star|s)$ can be queried for any given state $s \in \mathcal{S}$. Due to this requirement, the expert policy in online IL is often an *algorithm* (rather than a human demonstrator), which is hard-coded or based on additional computational resources, such as trajectory optimization (Pan et al., 2017a). The goal of IL is to learn a policy that can perform similar to, or better than, the expert policy.

[3] A similar equation holds for reparametrization (Grathwohl et al., 2018).

Rather than solving the stochastic nonconvex optimization directly, online IL solves an online learning problem with per-round cost in the nth iteration defined as

$$l_n(\pi) = \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi}[\tilde{c}] \tag{5.9}$$

where $\tilde{c} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a surrogate loss satisfying the following condition: For all $s \in \mathcal{S}$ and $\pi \in \Pi$, there exists a constant $C^{\pi^\star} > 0$ such that

$$C^{\pi^\star} \mathbb{E}_{\pi}[\tilde{c}] \geq \mathbb{E}_{\pi}[A^{\pi^\star}]. \tag{5.10}$$

By Lemma 5.2.1, this implies $J(\pi_n) \leq J(\pi^\star) + \frac{C^{\pi^\star}}{1-\gamma} l_n(\pi_n)$. Namely, in the nth iteration, online IL attempts to minimize an online upper-bound of $J(\pi_n)$.

DAGGER (Ross, Gordon, and Bagnell, 2011) chooses $\tilde{c}$ to be a loss function $\tilde{c}(s, a) = \mathbb{E}_{a^\star \sim \pi^\star | s}[m(a, a^\star)]$ that penalizes the difference between the learner's policy and the expert's policy, where m is some metric of space $\mathcal{A}$ (e.g., for a continuous action space Pan et al. (2017a) choose $m(a, a^\star) = \|a - a^\star\|^2$). More directly, AGGREVATTE simply chooses $\tilde{c}(s, a) = A^{\pi^\star}(s, a)$ (Ross and Bagnell, 2014); in this case, the policy learned with online IL can potentially outperform the expert policy.

First-order online IL methods operate by updating policies with mirror descent (5.4) with g_n as an estimate of

$$\nabla_\theta l_n(\pi_n) = \mathbb{E}_{d^{\pi_n}} \left(\nabla_\theta \mathbb{E}_{\pi}\right) [\tilde{c}]|_{\pi=\pi_n} \tag{5.11}$$

Similar to policy gradients, the implementation of (5.11) can be executed using either (5.7) or (5.8) (and with a control variate). One particular case of (5.11), with $\tilde{c} = A^{\pi^\star}$, is known as AGGREVATED (Sun et al., 2017),

$$\nabla_\theta l_n(\pi_n) = \mathbb{E}_{d^{\pi_n}} \left(\nabla_\theta \mathbb{E}_{\pi}\right) [A^{\pi^\star}]|_{\pi=\pi_n}. \tag{5.12}$$

Similarly, we can turn DAGGER into a first-order method, which we call DAGGERED, by using g_n as an estimate of the first-order oracle

$$\nabla_\theta l_n(\pi_n) = \mathbb{E}_{d^{\pi_n}} \left(\nabla_\theta \mathbb{E}_\pi\right) \mathbb{E}_{\pi^\star}[m]. \tag{5.13}$$

A comparison is summarized in Table 5.1.

5.4 Theoretical Comparison

With the first-order oracles defined, we now compare the performance and properties of performing mirror descent with policy gradient or imitation gradient. We will see that while both approaches share the same update rule in (5.4), the generated policies have different behaviors: using policy gradient generates a *monotonically* improving policy sequence, whereas using imitation gradient generates a policy sequence that improves *on average*. Although the techniques used in this section are not completely new in the optimization literature, we specialize the results to compare performance and to motivate LOKI in the next section. The proofs of this section are included in Section 5.B.

5.4.1 Policy Gradients

We analyze the performance of policy gradients with standard techniques from nonconvex analysis.

Proposition 5.4.1. *Let J be β-smooth and R_n be α_n-strongly convex with respect to norm $\|\cdot\|$. Assume $\mathbb{E}[g_n] = \nabla_\theta J(\pi_n)$. For $\eta_n \leq \frac{2\alpha_n}{\beta}$, it satisfies*

$$\begin{aligned}\mathbb{E}\left[J(\pi_{n+1})\right] \leq J(\pi_0) &+ \mathbb{E}\left[\sum_{n=1}^{N} \frac{2\eta_n}{\alpha_n} \|\nabla_\theta J(\pi_n) - g_n\|_*^2\right] \\ &+ \frac{1}{2}\mathbb{E}\left[\sum_{n=1}^{N} \left(-\alpha_n\eta_n + \frac{\beta\eta_n^2}{2}\right) \|\hat{\nabla}_\theta J(\pi_n)\|^2\right]\end{aligned}$$

where the expectation is due to randomness of sampling g_n, and $\hat{\nabla}_\theta J(\pi_n) \coloneqq \frac{1}{\eta_n}\left(\theta_n - P_{n,\nabla_\theta J(\pi_n)}(\theta_n)\right)$. *is a gradient surrogate.*

Proposition 5.4.1 shows that monotonic improvement can be made under proper smoothness assumptions if the step size is small and noise is comparably small with the gradient size. However, the final policy's performance is sensitive to the initial condition $J(\pi_0)$, which can be poor for a randomly initialized policy.

Proposition 5.4.1 also suggests that the size of the gradient $\|\hat{\nabla}_\theta J(\pi_n)\|^2$ does not converge to zero on average. Instead, it converges to a size proportional to the sampling noise of policy gradient estimates due to the linear dependency of $\frac{2\eta_n}{\alpha_n}\|\nabla_\theta J(\pi_n) - g_n\|_*^2$ on η_n. This phenomenon is also mentioned by Ghadimi, Lan, and Zhang, 2016. We note that this pessimistic result is because the prox-map (5.5) is nonlinear in g_n for general R_n and Θ. However, when R_n is quadratic and Θ is unconstrained, the convergence of $\|\hat{\nabla}_\theta J(\pi_n)\|^2$ to zero on average can be guaranteed (see Section 5.B.1 for a discussion).

5.4.2 Imitation Gradients

While applying mirror descent with a policy gradient can generate a monotonically improving policy sequence, applying the same algorithm with an imitation gradient yields a different behavior. The result is summarized below, which is a restatement of Ross and Bagnell, 2014, Theorem 2.1, but is specialized for mirror descent.

Proposition 5.4.2. *Assume l_n is σ-strongly convex with respect to R_n.*[4] *Assume $\mathbb{E}[g_n] = \nabla_\theta l_n(\pi_n)$ and $\|g_n\|_* \leq G < \infty$ almost surely. For $\eta_n = \frac{1}{\hat{\sigma} n}$ with $\hat{\sigma} \leq \sigma$, it holds*

$$\frac{1}{N}\mathbb{E}\left[\sum_{n=1}^{N} J(\pi_n)\right] \leq J(\pi^\star) + \frac{C^{\pi^\star}}{1-\gamma}\left(\epsilon_{\textit{class}} + \epsilon_{\textit{regret}}\right)$$

where the expectation is due to randomness of sampling g_n, $\epsilon_{\textit{class}} = \sup_{\{\pi_n\}} \inf_{\pi\in\Pi} \frac{1}{N}\sum_{n=1}^{N} l_n(\pi)$ and $\epsilon_{\textit{regret}} = \frac{G^2(\log N + 1)}{2\hat{\sigma} N}$.

[4] A function f is said to be σ-strongly convex with respect to R on a set $\mathcal{K}$ if for all $x, y \in \mathcal{K}$, $f(x) \geq f(y) + \langle \nabla f(y), x - y\rangle + \sigma D_R(x\|y)$.

Proposition 5.4.2 is based on the assumption that l_n is strongly convex, which can be verified for certain problems (Cheng and Boots, 2018). Consequently, Proposition 5.4.2 shows that the performance of the policy sequence on average can converge close to the expert's performance $J(\pi^\star)$, with additional error that is proportional to ϵ_{class} and ϵ_{regret}.

ϵ_{regret} is an upper bound of the average regret, which is less than $\tilde{O}(\frac{1}{N})$ for a *large* enough step size.[5] This characteristic is in contrast to policy gradient, which requires *small* enough step sizes to guarantee local improvement.

ϵ_{class} measures the expressiveness of the policy class Π. It can be *negative* if there is a policy in Π that outperforms the expert policy $\pi^\star$ in terms of $\tilde{c}$. However, since online IL attempts to minimize an online upper bound of the accumulated cost through a surrogate loss $\tilde{c}$, the policy learned with imitation gradients in general cannot be better than performing one-step policy improvement from the expert policy (Cheng and Boots, 2018; Ross and Bagnell, 2014). Therefore, when the expert is suboptimal, the reduction from nonconvex optimization to online convex optimization can lead to suboptimal policies.

Finally, we note that updating policies with imitation gradients does not necessarily generate a monotonically improving policy sequence, even for deterministic problems; whether the policy improves monotonically is completely problem dependent (Cheng and Boots, 2018). Without going into details, we can see this by comparing policy gradient in (5.6) and the special case of imitation gradient in (5.12). By Lemma 5.3, we see that

$$\mathbb{E}_{d^{\pi_n}}\left(\nabla_\theta \mathbb{E}_\pi\right)\left[A^{\pi_n}\right] = \left(\nabla_\theta \mathbb{E}_{d^\pi}\right)\mathbb{E}_{\pi_n}\left[A^{\pi^\star}\right] + \mathbb{E}_{d^{\pi_n}}\left(\nabla_\theta \mathbb{E}_\pi\right)\left[A^{\pi^\star}\right].$$

Therefore, even with $\tilde{c} = A^{\pi^\star}$, the negative of the direction in (5.12) is not necessarily a descent direction; namely applying (5.12) to update the policy is not guaranteed to improve the policy performance locally.

[5]The step size should be large enough to guarantee $\tilde{O}(\frac{1}{N})$ convergence, where $\tilde{O}$ denotes Big-O but omitting $\log$ dependency. However, it should be bounded since $\epsilon_{\text{regret}} = \Theta\left(\frac{1}{\tilde{\sigma}}\right)$.

Algorithm 1 LOKI

Parameters: d, N_m, N_M
Input: $\pi^\star$

Sample K with probability in (5.15).
for $t = 1 \dots K$ **do** $\{$ # Imitation Phase$\}$
  Collect data $\mathcal{D}_n$ by executing π_n
  Query g_n from (5.11) using $\pi^\star$
  Update π_n by mirror descent (5.4) with g_n
  Update advantage function estimate $\hat{A}^{\pi_n}$ by $\mathcal{D}_n$
end for
for $t = K+1 \dots$ **do** $\{$ # Reinforcement Phase$\}$
  Collect data $\mathcal{D}_n$ by executing π_n.
  Query g_n from (5.6) f using $\hat{A}^{\pi_n}$
  Update π_n by mirror descent (5.4) with g_n
  Update advantage function estimate $\hat{A}^{\pi_n}$ by $\mathcal{D}_n$
end for

5.5 Imitate-Then-Reinforce

To combine the benefits from RL and IL, we propose a simple randomized algorithm LOKI: first perform K steps of mirror descent with imitation gradient and then switch to policy gradient for the rest of the steps. Despite the algorithm's simplicity, we show that, when K is appropriately randomized, running LOKI has similar performance to performing policy gradient steps directly from the expert policy.

5.5.1 Algorithm: LOKI

The algorithm LOKI is summarized in Algorithm 1. The algorithm is composed of two phases: an imitation phase and a reinforcement phase. In addition to learning rates, LOKI receives three hyperparameters (d, N_m, N_M) which determine the probability of random switching at time K. As shown in the next section, these three hyperparameters can be selected fairly simply.

Imitation Phase Before learning, LOKI first randomly samples a number $K \in [N_m, N_M]$ according to the prescribed probability distribution (5.15). Then it performs K steps of

mirror descent with imitation gradient. In our implementation, we set

$$\mathbb{E}_{\pi}[\tilde{c}] = \mathrm{KL}(\pi^{\star}||\pi), \tag{5.14}$$

which is the KL-divergence between the two policies. (The reverse KL-divergence can also be used.) It can be easily shown that a proper constant $C^{\pi^{\star}}$ exists satisfying the requirement of $\tilde{c}$ in (5.10) (Gibbs and Su, 2002). While using (5.14) does not guarantee learning a policy that outperforms the expert due to $\epsilon_{\text{class}} \geq 0$, with another reinforcement phase available, the imitation phase of LOKI is only designed to quickly bring the initial policy closer to the expert policy. Compared with choosing $\tilde{c} = A^{\pi^{\star}}$ as in AGGREVATED, one benefit of choosing $\mathrm{KL}(\pi^{\star}||\pi)$ (or its variants, e.g. $\|a - a^{\star}\|^2$) is that it does not require learning a value function estimator. In addition, the imitation gradient can be calculated through *reparametrization* instead of a likelihood-ratio (Tucker et al., 2017), as now $\tilde{c}$ is presented as a differentiable function in a. Consequently, the sampling variance of imitation gradient can be significantly reduced by using multiple samples of $a \sim \pi_n$ (with a single query from the expert policy) and then performing averaging.

Reinforcement Phase After the imitation phase, LOKI switches to the reinforcement phase. At this point, the policy π_K is much closer to the expert policy than the initial policy π_0. In addition, an estimate of A_{π_K} is also available. Because the learner's policies were applied to collect data in the previous *online* imitation phase, A^{π_n} can already be updated accordingly, for example, by minimizing TD error. Compared with other warm-start techniques, LOKI can learn *both* the policy and the advantage estimator in the imitation phase.

5.5.2 Analysis

We now present the theoretical properties of LOKI. The analysis is composed of two steps. First, we show the performance of $J(\pi_K)$ in Theorem 5.5.1, a generalization of Propo-

sition 5.4.2 to consider the effects of non-uniform random sampling. Next, combining Theorem 5.5.1 and Proposition 5.4.1, we show the performance of LOKI in Theorem 5.5.2. The proofs are given in Section 5.C.

Theorem 5.5.1. *Let $d \geq 0$, $N_m \geq 1$, and $N_M \geq 2N_m$. Let $K \in [N_m, N_M]$ be a discrete random variable such that*

$$P(K = n) = \frac{n^d}{\sum_{m=N_m}^{N_M} m^d}. \tag{5.15}$$

Suppose l_n is σ-strongly convex with respect to R_n, $\mathbb{E}[g_n] = \nabla_\theta l_n(\pi_n)$, and $\|g_n\|^* \leq G < \infty$ almost surely. Let $\{\pi_n\}$ be generated by running mirror descent with step size $\eta_n = n^d/\hat{\sigma}\sum_{m=1}^{n} m^d$. For $\hat{\sigma} \leq \sigma$, it holds that

$$\mathbb{E}\left[J(\pi_K)\right] \leq J(\pi^\star) + \Delta,$$

where the expectation is due to sampling K and g_n, $\Delta = \frac{C^{\pi^\star}}{1-\gamma}\left(\epsilon^w_{class} + 2^{-d}\hat{\sigma}D_{\mathcal{R}} + G^2 C_{N_M}/\hat{\sigma}N_M\right)$, $D_{\mathcal{R}} = \sup_{R\in\mathcal{R}}\sup_{\pi,\pi'\in\Pi} D_R(\pi'||\pi)$, $\epsilon^w_{class} \coloneqq \sup_{\{w_n\},\{\pi_n\}}\inf_{\pi\in\Pi}\frac{\sum_{n=1}^{N} w_n l_n(\pi)}{\sum_{n=1}^{N} w_n}$, and

$$C_{N_M} = \begin{cases} \log(N_M) + 1, & \text{if } d = 0 \\ \frac{8d}{3}\exp\left(\frac{d}{N_M}\right), & \text{if } d \geq 1 \end{cases}$$

Suppose $N_M \gg d$. Theorem 5.5.1 says that the performance of $J(\pi_K)$ in expectation converges to $J(\pi^\star)$ in a rate of $\tilde{O}(d/N_M)$ when a proper step size is selected. In addition to the convergence rate, we notice that the performance gap between $J(\pi^\star)$ and $J(\pi_K)$ is bounded by $O(\epsilon^w_{\text{class}} + 2^{-d}D_{\mathcal{R}})$. $\epsilon^w_{\text{class}}$ is a weighted version of the expressiveness measure of policy class Π in Proposition 5.4.2, which can be made small if Π is rich enough with respect to the *suboptimal* expert policy. $D_{\mathcal{R}}$ measures the size of the decision space with respect to the class of regularization functions $\mathcal{R}$ that the learner uses in mirror descent. The dependency on $D_{\mathcal{R}}$ is because Theorem 5.5.1 performs a suffix random sampling with

$N_m > 0$. While the presence of $D_{\mathcal{R}}$ increases the gap, its influence can easily made small with a slightly large d due to the factor 2^{-d}.

In summary, due to the sublinear convergence rate of IL, N_M does not need to be large (say less than 100) as long as $N_M \gg d$; on the other hand, due to the 2^d factor, d is also small (say less than 5) as long as it is large enough to cancel out the effects of $D_{\mathcal{R}}$. Finally, we note that, like Proposition 5.4.2, Theorem 5.5.1 encourages using larger step sizes, which can further boost the convergence of the policy in the imitation phase of LOKI.

Given Proposition 5.4.1 and Theorem 5.5.1, now it is fairly easy to understand the performance of LOKI.

Theorem 5.5.2. *Running* LOKI *holds that*

$$\mathbb{E}\left[J(\pi_N)\right] \leq J(\pi^\star) + \Delta + \mathbb{E}\left[\sum_{n=K+1}^{N} \frac{2\eta_n}{\alpha_n} \|\nabla_\theta J(\pi_n) - g_n\|_*^2\right] + \frac{1}{2}\mathbb{E}\left[\sum_{n=K+1}^{N} \left(-\alpha_n\eta_n + \frac{\beta\eta_n^2}{2}\right) \|\hat{\nabla}_\theta J(\pi_n)\|^2\right],$$

where the expectation is due to sampling g_n and K.

Firstly, Theorem 5.5.2 shows that π_N can perform better than the expect policy $\pi^\star$, and, in fact, it converges to a locally optimal policy on average under the same assumption as in Proposition 5.4.1. Compared with to running policy gradient steps directly from the expert policy, running LOKI introduces an additional gap $O(\Delta + K\|\hat{\nabla}_\theta J(\pi)\|^2)$. However, as discussed previously, Δ and $K \leq N_M \ll N$ are reasonably small, for usual N in RL. Therefore, performing LOKI almost has the same effect as using the expert policy as the initial condition, which is the best we can hope for when having access to an expert policy.

We can also compare LOKI with performing usual policy gradient updates from a randomly initialized policy. The performance difference can be easily shown as $O(J(\pi^\star) - J(\pi_0) + \Delta + K\|\hat{\nabla}_\theta J(\pi)\|^2)$. Therefore, if performing K steps of policy gradient from π_0 gives a policy with performance worse than $J(\pi^\star) + \Delta$, then LOKI is favorable.

5.6 Related Work

We compare LOKI with some recent attempts to incorporate the loss information c of RL into IL so that it can learn a policy that outperforms the expert policy. As discussed in Section 5.4, when $\tilde{c} = A^{\pi^\star}$, AGGREVATE(D) can potentially learn a policy that is better than the expert policy (Ross and Bagnell, 2014; Sun et al., 2017). However, implementing AGGREVATE(D) exactly as suggested by theory can be difficult and inefficient in practice. On the one hand, while $A^{\pi^\star}$ can be learned off-policy using samples collected by running the expert policy, usually the estimator quality is unsatisfactory due to covariate shift. On the other hand, if $A^{\pi^\star}$ is learned on-policy, it requires restarting the system from any state, or requires performing $\frac{1}{1-\gamma}$-times more iterations to achieve the same convergence rate as other choices of $\tilde{c}$ such as $\mathrm{KL}(\pi^\star||\pi)$ in LOKI; both of which are impractical for usual RL problems.

Recently, Sun, Bagnell, and Boots (2018) proposed THOR (Truncated HORizon policy search) which solves a truncated RL problem with the expert's value function as the terminal loss to alleviate the strong dependency of AGGREVATED on the quality of $A^{\pi^\star}$. Their algorithm uses an H-step truncated advantage function defined as

$$A^{H,\pi^\star}_{\pi_n,t} = \mathbb{E}_{\rho_{\pi_n}}[\sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} c(s_\tau, a_\tau) + \gamma^H V_{\pi^\star}(s_{t+H}) - V_{\pi^\star}(s_t)].$$

While empirically the authors show that the learned policy can improve over the expert policy, the theoretical properties of THOR remain somewhat unclear.[6] In addition, THOR is more convoluted to implement and relies on multiple advantage function estimators. By contrast, LOKI has stronger theoretical guarantees, while being straightforward to implement with off-the-shelf learning algorithms.

Finally, we compare LOKI with LOLS (Locally Optimal Learning to Search), proposed

[6]The algorithm actually implemented by Sun, Bagnell, and Boots (2018) does not solve precisely the same problem analyzed in theory.

by Chang et al., 2015. LOLS is an online IL algorithm which sets $\tilde{c} = Q^{\hat{\pi}_n^\lambda}$, where $\lambda \in [0, 1]$ and $\hat{\pi}_n^\lambda$ is a mixed policy that at each time step chooses to run the current policy π_n with probability $1 - \lambda$ and the expert policy $\pi^\star$ with probability λ. Like AGGREVATED, LOLS suffers from the impractical requirement of estimating $Q^{\hat{\pi}_n^\lambda}$, which relies on the state resetting assumption.

Here we show that such difficulty can be addressed by using the mirror descent framework with g_n as an estimate of $\nabla_\theta l_n^\lambda(\pi_n)$, where $l_n^\lambda(\pi) := \mathbb{E}_{d^{\pi_n}}\mathbb{E}_\pi[(1-\lambda)A^{\pi_n} + \lambda A^{\pi^\star}]$. That is, the first-order oracle is simply a convex combination of policy gradient and AGGREVATED gradient. We call such linear combination SLOLS (simple LOLS) and we show it has the same performance guarantee as LOLS.

Theorem 5.6.1. *Under the same assumption in Proposition 5.4.2, running* SLOLS *generates a policy sequence, with randomness due to sampling g_n, satisfying*

$$\frac{1}{N}\mathbb{E}\left[\sum_{n=1}^{N} J(\pi_n) - \left((1-\lambda)J^*_{\pi_n} + \lambda J(\pi^\star)\right)\right] \leq \frac{\epsilon^\lambda_{class} + \epsilon^\lambda_{regret}}{1-\gamma}$$

where $J^*_{\pi_n} = \min_{\pi\in\Pi} \mathbb{E}_{d^{\pi_n}}\mathbb{E}_\pi[Q^{\pi_n}] =: \mathbb{E}_{d^{\pi_n}}[V^*_{\pi_n}]$ *and* $\epsilon^\lambda_{class} = \min_{\pi\in\Pi} \frac{1}{N}(\sum_{n=1}^N \mathbb{E}_{d^{\pi_n}}\mathbb{E}_\pi[(1-\lambda)Q^{\pi_n} + \lambda Q_{\pi^\star}]) - \frac{1}{N}(\sum_{n=1}^N \mathbb{E}_{d^{\pi_n}}[(1-\lambda)V^*_{\pi_n} + \lambda V_{\pi^\star}])$.

In fact, the performance in Theorem 5.6.1 is actually a lower bound of Theorem 3 in (Chang et al., 2015).[7] Theorem 5.6.1 says that on average π_n has performance between the expert policy $J(\pi^\star)$ and the intermediate cost $J^*_{\pi_n}$, as long as $\epsilon^\lambda_{\text{class}}$ is small (i.e., there exists a single policy in Π that is better than the expert policy or the local improvement from any policy in Π). However, due to the presence of $\epsilon^\lambda_{\text{class}}$, despite $J^*_{\pi_n} \leq J(\pi_n)$, it is not guaranteed that $J^*_{\pi_n} \leq J(\pi^\star)$. As in Chang et al., 2015, either LOLS or SLOLS can necessarily perform on average better than the expert policy $\pi^\star$. Finally, we note that recently both Nair et al. (2017) and Rajeswaran et al. (2017) propose a scheme similar to SLOLS, but with the AGGREVATE(D) gradient computed using offline batch data collected by the

[7] The main difference is due to technicalities. In Chang et al., 2015, $\epsilon^\lambda_{\text{class}}$ is compared with a time-varying policy.

expert policy. However, there is no theoretical analysis of this algorithm's performance.

5.7 Experiments

We evaluate LOKI on several robotic control tasks from OpenAI Gym (Brockman et al., 2016) with the DART physics engine (Lee et al., 2018a)[8] and compare it with several baselines: TRPO (Schulman et al., 2015b), TRPO from expert, DAGGERED (the first-order version of DAGGER (Ross, Gordon, and Bagnell, 2011) in (5.13)), SLOLS (Section 5.6), and THOR (Sun, Bagnell, and Boots, 2018).

5.7.1 Tasks

We consider the following tasks. In all tasks, the discount factor of the RL problem is set to $\gamma = 0.99$. The details of each task are specified in Table 10.2 in Section 5.A.

Inverted Pendulum This is a classic control problem, and its goal is to swing up an pendulum and to keep it balanced in a upright posture. The difficulty of this task is that the pendulum cannot be swung up directly due to a torque limit.

Locomotion The goal of these tasks (Hopper, 2D Walker, and 3D Walker) is to control a walker to move forward as quickly as possible without falling down. In Hopper, the walker is a monoped, which is subjected to significant contact discontinuities, whereas the walkers in the other tasks are bipeds. In 2D Walker, the agent is constrained to a plane to simplify balancing.

Robot Manipulator In the Reacher task, a 5-DOF (degrees-of-freedom) arm is controlled to reach a random target position in 3D space. The reward consists of the negative distance to the target point from the finger tip plus a control magnitude penalty. The actions correspond to the torques applied to the 5 joints.

[8] The environments are defined in DartEnv, hosted at https://github.com/DartEnv.

5.7.2 Algorithms

We compare five algorithms (LOKI, TRPO, DAGGERED, THOR, SLOLS) and the idealistic setup of performing policy gradient steps directly from the expert policy (Ideal). To facilitate a fair comparison, all the algorithms are implemented based on a publicly available TRPO implementation (Dhariwal et al., 2017). Furthermore, they share the same parameters except for those that are unique to each algorithm as listed in Table 10.2 in Section 5.A. The experimental results averaged across 25 random seeds are reported in Section 5.7.3.

Policy and Value Networks Feed-forward neural networks are used to construct the policy networks and the value networks in all the tasks (both have two hidden layers and 32 tanh units per layer). We consider Gaussian stochastic policies, i.e. for any state $s \in \mathcal{S}$, $\pi(a|s)$ is Gaussian distributed. The mean of the Gaussian $\pi(a|s)$, as a function of state, is modeled by the policy network, and the covariance matrix of Gaussian is restricted to be diagonal and independent of state. The policy networks and the value function networks are initialized randomly, except for the ideal setup (TRPO from expert), which is initialized as the expert.

Expert Policy The same sub-optimal expert is used by all algorithms (LOKI, DAGGERED, SLOLS, and THOR). It is obtained by running TRPO and stopping it before convergence. The estimate of the expert value function $V_{\pi^\star}$ (required by SLOLS and THOR) is learned by minimizing the sum of squared TD(0) error on a large separately collected set of demonstrations of this expert. The final explained variance for all the tasks is more than 0.97 (see Section 5.A).

First-Order Oracles The on-policy advantage A^{π_n} in the first-order oracles for TRPO, SLOLS, and LOKI (in the reinforcement phase) is implemented using an on-policy value function estimator and Generalized Advantage Estimator (GAE) (Schulman et al., 2015a). For DAGGERED and the imitation phase of LOKI, the first-order oracle is calculated us-

ing (5.14). For SLOLS, we use the estimate $A^{\pi^\star}(s_t, a_t) \approx c(s_t, a_t) + \gamma \hat{V}^{\pi^\star}(s_{t+1}) - \hat{V}^{\pi^\star}(s_t)$. And for THOR, $A^{H,\pi^\star}_{\pi_n,t}$ of the truncated-horizon problem is approximated by Monte-Carlo samples with an on-policy value function baseline estimated by regressing on these Monte-Carlo samples. Therefore, for all methods, an on-policy component is used in constructing the first-order oracle. The exponential weighting in GAE is 0.98; the mixing coefficient λ in SLOLS is 0.5; N_M in LOKI is reported in Table 10.2 in Section 5.A, and $N_m = \left\lfloor \frac{1}{2} N_M \right\rfloor$, and $d = 3$.

Mirror Descent After receiving an update direction g_n from the first-order oracle, a KL-divergence-based trust region is specified. This is equivalent to setting the strictly convex function R_n in mirror descent to $\frac{1}{2}\theta^\top F(\theta_n)\theta$ and choosing a proper learning rate. In our experiments, a larger KL-divergence limit (0.1) is selected for imitation gradient (5.14) (in DAGGERED and in the imitation phase of LOKI), and a smaller one (0.01) is set for all other algorithms. This decision follows the guideline provided by the theoretical analysis in Section 5.3.2 and is because of the low variance in calculating the gradient of (5.14). Empirically, we observe using the larger KL-divergence limit with policy gradient led to high variance and instability.

5.7.3 Experimental Results

We report the performance of these algorithms on various tasks in Fig. 5.1. The performance is measured by the accumulated *rewards*, which are directly provided by OpenAI Gym.

We first establish the performance of two baselines, which represent standard RL (TRPO) and standard IL (DAGGERED). TRPO is able to achieve considerable and almost monotonic improvement from a randomly initialized policy. DAGGERED reaches the performance of the suboptimal policy in a relatively very small number of iterations, e.g. 15 iterations in 2D Walker, in which the suboptimal policy to imitate is TRPO at iteration 100. However, it

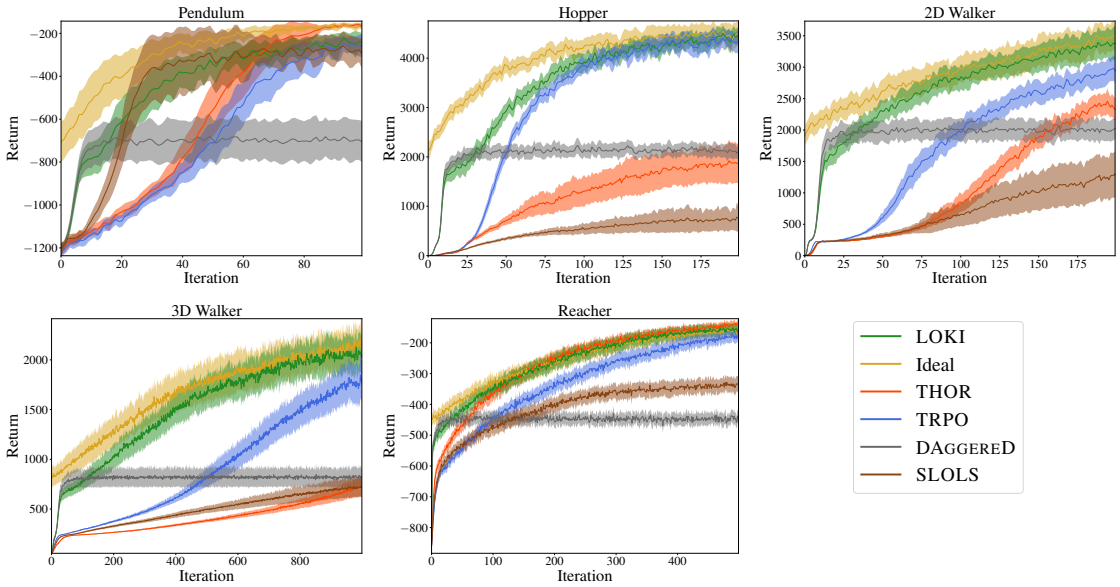


Figure 5.1: Learning curves. Shaded regions correspond to $\pm\frac{1}{2}$-standard deviation.

fails to outperform the suboptimal expert.

Then, we evaluate the proposed algorithm LOKI and Ideal, the performance of which we wish to achieve in theory. LOKI consistently enjoys the best of both TRPO and DAGGERED: it improves as fast as DAGGERED at the beginning, keeps improving, and then finally matches the performance of Ideal after transitioning into the reinforcement phase. Interestingly, the on-policy value function learned, though not used, in the imitation phase helps LOKI transition from imitation phase to reinforcement phase smoothly.

Lastly, we compare LOKI to the two other baselines (SLOLS and THOR) that combine RL and IL. LOKI outperforms these two baselines by a considerably large margin in Hopper, 2D Walker, and 3D Walker; but surprisingly, the performance of SLOLS and THOR are inferior even to TRPO on these tasks. The main reason is that the first-order oracles of both methods is based on an estimated expert value function $\hat{V}^{\pi^\star}$. As $\hat{V}^{\pi^\star}$ is only regressed on the data collected by running the expert policy, large covariate shift error could happen if the dimension of the state and action spaces are high, or if the uncontrolled system is complex or unstable. For example, in the low-dimensional Pendulum task and the simple Reacher task, the expert value function can generalize better. As a result, in these two cases, LOLS

and THOR achieve super-expert performance. However, in more complex tasks, where the effects of covariant shift amplifies exponentially with the dimension of the state space, THOR and SLOLS start to suffer from the inaccuracy of $\hat{V}^{\pi^\star}$, as illustrated in the 2D Walker and 3D Walker tasks.

5.8 Conclusion

We present a simple, elegant algorithm, LOKI, that combines the best properties of RL and IL. Theoretically, we show that, by randomizing the switching time, LOKI can perform as if running policy gradient steps directly from the expert policy. Empirically, LOKI demonstrates superior performance compared with the expert policy and more complicated algorithms that attempt to combine RL and IL.

5.A Task Details

Table 5.2: Experiment Details

	Pendulum	Hopper	2D Walker	3D Walker	Reacher
Observation space dimension	3	11	17	41	21
Action space dimension	1	3	6	15	5
Number of samples per iteration	4k	16k	16k	16k	40k
Number of iterations	100	200	200	1000	500
Number of TRPO iterations for expert	50	50	100	500	100
Upper limit of number of imitation steps of LOKI	10	20	25	50	25
Truncated horizon of THOR	40	40	250	250	250

The expert value estimator $\hat{V}^{\pi^\star}$ needed by SLOLS and THOR were trained on a large set of samples (50 times the number of samples used in each batch in the later policy learning), and the final average TD error are: Pendulum (0.972), Hopper (0.989), 2D Walker (0.975), 3D Walker (0.983), and Reacher (0.973), measured in terms of explained variance, which is defined as 1- (variance of error / variance of prediction).

5.B Proof of Section 5.4

5.B.1 Proof of Proposition 5.4.1

To prove Proposition 5.4.1, we first prove a useful Lemma 5.B.1.

Lemma 5.B.1. *Let $\mathcal{K}$ be a convex set. Let $h = \mathbb{E}[g]$. Suppose R is α-strongly convex with respect to norm $\|\cdot\|$.*

$$y = \arg\min_{z \in \mathcal{K}} \langle g, z\rangle + \frac{1}{\eta} d^R(z||x) =: P_{g,\eta}(x)$$

where η satisfies that $-\alpha\eta + \frac{\beta\eta^2}{2} \leq 0$. Then it holds

$$\mathbb{E}[\langle h, y - x\rangle + \frac{\beta}{2}\|x - y\|^2] \leq \frac{1}{2}\left(-\alpha\eta + \frac{\beta\eta^2}{2}\right)\mathbb{E}\left[\|H\|^2\right] + \frac{2\eta}{\alpha}\mathbb{E}\left[\|g - h\|_*^2\right]$$

where $H = \frac{1}{\eta}(x - P_{h,\eta}(x))$. In particular, if $\|\cdot\| = \|\cdot\|_W$ for some positive definite matrix W, R is quadratic, and $\mathcal{K}$ is Euclidean space,

$$\mathbb{E}[\langle h, y - x\rangle + \frac{\beta}{2}\|x - y\|^2] \leq \left(-\alpha\eta + \frac{\beta\eta^2}{2}\right)\mathbb{E}\left[\|H\|^2\right] + \frac{\beta\eta^2}{2}\mathbb{E}[\|H - G\|^2]$$

Proof. Let $G = \frac{1}{\eta}(x - P_{g,\eta}(x))$. First we show for the special case (i.e. suppose $R(x) = \frac{1}{2}\langle x, Mx\rangle$ for some positive definite matrix M, and therefore $G = M^{-1}g$ and $H = M^{-1}h$).

$$\mathbb{E}[\langle h, y - x\rangle] = -\eta\mathbb{E}[\langle h, G\rangle] = -\eta\mathbb{E}[\langle h, H\rangle] \leq -\alpha\eta\|H\|^2$$

and because g is unbiased,

$$\mathbb{E}\left[\frac{\beta}{2}\|x - y\|^2\right] = \mathbb{E}\left[\frac{\eta^2\beta}{2}\|H\|^2 + \frac{\eta^2\beta}{2}\|G - H\|^2\right]$$

For general setups, we first separate the term into two parts

$$\langle h, y - x \rangle = \langle g, y - x \rangle + \langle h - g, y - x \rangle$$

For the first term, we use the optimality condition

$$\left\langle g + \frac{1}{\eta}\nabla R(y) - \frac{1}{\eta}\nabla R(x), z - y \right\rangle \geq 0, \quad \forall z \in \mathcal{K}$$

which implies

$$\langle g, x - y \rangle \geq \frac{1}{\eta} \langle \nabla R(y) - \nabla R(x), y - x \rangle \geq \frac{\alpha}{\eta}\|x - y\|^2$$

Therefore, we can bound the first term by

$$\langle g, y - x \rangle \leq -\frac{\alpha}{\eta}\|x - y\|^2 = -\alpha\eta\|G\|^2$$

On the other hand, for the second term, we first write

$$\begin{aligned} \langle h - g, y - x \rangle &= -\eta \langle h - g, G \rangle \\ &= -\eta \langle h - g, H \rangle + \eta \langle h - g, H - G \rangle \end{aligned}$$

and we show that

$$\langle h - g, H - G \rangle \leq \|h - g\|_* \|H - G\| \leq \frac{\|h - g\|_*^2}{\alpha} \tag{5.16}$$

This can be proved by Legendre transform:

$$\begin{aligned} P_{g,\eta}(x) &= \arg\min_{z\in\mathcal{K}} \langle g, z\rangle + \frac{1}{\eta} d^R(z||x) \\ &= \arg\min_{z\in\mathcal{K}} \left\langle g - \frac{1}{\eta}\nabla R(x), z \right\rangle + \frac{1}{\eta} R(z) \\ &= \nabla \left(\frac{1}{\eta} R\right)^* \left(\frac{1}{\eta}\nabla R(x) - g\right) \end{aligned}$$

Because $\frac{1}{\eta}R$ is $\frac{\alpha}{\eta}$-strongly convex with respect to norm $\|\cdot\|$, $\left(\frac{1}{\eta}R\right)^*$ is $\frac{\eta}{\alpha}$-smooth with respect to norm $\|\cdot\|_*$, we have

$$\|H - G\| \leq \frac{1}{\eta}\frac{\eta}{\alpha}\|g - h\|_* = \frac{1}{\alpha}\|g - h\|_*$$

which proves (5.16). Putting everything together, we have

$$\begin{aligned} &\mathbb{E}[\langle h, y - x\rangle + \frac{\beta}{2}\|x - y\|^2] \\ &\leq \mathbb{E}\left[\left(-\alpha\eta + \frac{\beta\eta^2}{2}\right)\|G\|^2\right] + \mathbb{E}\left[-\eta\langle h - g, H\rangle + \frac{\eta}{\alpha}\|g - h\|_*^2\right] \\ &= \mathbb{E}\left[\left(-\alpha\eta + \frac{\beta\eta^2}{2}\right)\|G\|^2\right] + \mathbb{E}\left[\frac{\eta}{\alpha}\|g - h\|_*^2\right] \end{aligned}$$

Because

$$\|H\|^2 \leq 2\|G\|^2 + 2\|H - G\|^2 \leq 2\|G\|^2 + \frac{2}{\alpha^2}\|h - g\|_*^2$$

it holds that

$$\begin{aligned} &\mathbb{E}[\langle h, y - x\rangle + \frac{\beta}{2}\|x - y\|^2] \\ &\leq \frac{1}{2}\left(-\alpha\eta + \frac{\beta\eta^2}{2}\right)\mathbb{E}\left[\|H\|^2\right] + \frac{2\eta}{\alpha}\mathbb{E}\left[\|g - h\|_*^2\right] \end{aligned}$$

■

Proof of Proposition 5.4.1 We apply Lemma 5.B.1: By smoothness of J,

$$\begin{aligned}\mathbb{E}\left[J(\pi_{n+1})\right] - J(\pi_n) &\leq \mathbb{E}\left[\langle \nabla J(\pi_n), \theta_{n+1} - \theta_n\rangle + \frac{\beta}{2}\|\theta_{n+1} - \theta_n\|^2\right] \\ &\leq \frac{1}{2}\left(-\alpha_n\eta_n + \frac{\beta\eta_n^2}{2}\right)\mathbb{E}\left[\|\hat{\nabla}_\theta J(\pi_n)\|^2\right] + \frac{2\eta_n}{\alpha_n}\|\nabla_\theta J(\pi_n) - g_n\|_*^2\end{aligned}$$

This proves the statement in Proposition 5.4.1. We note that, in the above step, the general result of Lemma 5.B.1. For the special case Lemma 5.B.1, we would recover the usual convergence property of stochastic smooth nonconvex optimization, which shows on average convergence to stationary points in expectation.

5.B.2 Proof of Proposition 5.4.2

We use a well-know result of mirror descent, whose proof can be found e.g. in (Juditsky and Nemirovski, 2011).

Lemma 5.B.2. *Let $\mathcal{K}$ be a convex set. Suppose R is α-strongly convex with respect to norm $\|\cdot\|$. Let g be a vector in some Euclidean space and let*

$$y = \arg\min_{z\in\mathcal{K}} \langle g, z\rangle + \frac{1}{\eta}d^R(z||x) = P_{g,\eta}(x)$$

Then for all $z \in \mathcal{K}$

$$\eta\langle g, x - z\rangle \leq D_R(z||x) - D_R(z||y) + \frac{\eta^2}{2}\|g\|_*^2$$

Next we prove a lemma of performing online mirror descent with weighted cost. While weighting it not required in proving Proposition 5.4.2, it will be useful to prove Theorem 5.5.2 later in Section 5.C.

Lemma 5.B.3. *Let f_n be σ-strongly convex with respect to some strictly convex function*

R_n, *i.e.*

$$f_n(x) \geq f_n(y) + \langle \nabla f_n(y), x - y \rangle + \sigma d^{R_n}(x||y)$$

and let $\{w_n\}_{n=1}^N$ *be a sequence of positive numbers. Consider the update rule*

$$x_{n+1} = \arg\min_{z \in \mathcal{K}} \langle w_n g_n, x \rangle + \frac{1}{\eta_n} d^{R_n}(z||x_n)$$

where $g_n = \nabla f_n(x_n)$ *and* $\eta_n = \frac{1}{\hat{\sigma} \sum_{m=1}^n w_m}$. *Suppose* $\hat{\sigma} \leq \sigma$. *Then for all* $x^* \in \mathcal{K}$, $N \geq M \geq 1$, *it holds that*

$$\sum_{n=M}^N w_n f_n(x_n) - w_n f_n(x^*) \leq \hat{\sigma} d^{R_M}(x^*||x_M) \sum_{n=1}^{M-1} w_n + \frac{1}{2\hat{\sigma}} \sum_{n=1}^N \frac{w_n^2 \|g_n\|_*^2}{\sum_{m=1}^n w_m}$$

Proof. The proof is straight forward by strong convexity of f_n and Lemma 10.F.1.

$$\begin{aligned}
&\sum_{n=M}^N w_n (f_n(x_n) - f_n(x^*)) \\
&\leq \sum_{n=M}^N w_n \left(\langle g_n, x_n - x^* \rangle - \sigma d^{R_n}(x^*||x_n) \right) && (\sigma\text{-strong convexity}) \\
&\leq \sum_{n=M}^N \frac{1}{\eta_n} d^{R_n}(x^*||x_n) - \frac{1}{\eta_n} d^{R_n}(x^*||x_{n+1}) - w_n \sigma d^{R_n}(x^*||x_n) + \frac{w_n^2 \eta_n}{2} \|g_n\|_*^2 && (\text{Lemma 10.F.1}) \\
&\leq \frac{d^{R_M}(x^*||x_M)}{\eta_{M-1}} + \sum_{n=M}^N \left(\frac{1}{\eta_n} - \frac{1}{\eta_{n-1}} - w_n \sigma \right) d^{R_n}(x^*||x_n) + \frac{w_n^2 \eta_n}{2} \|g_n\|_*^2 && (\text{We define } \frac{1}{\eta_0} = 0) \\
&= \hat{\sigma} d^{R_M}(x^*||x_M) \sum_{n=1}^{M-1} w_n + \sum_{n=1}^N (w_n \hat{\sigma} - w_n \sigma) d^{R_n}(x^*||x_n) + \frac{1}{2\hat{\sigma}} \sum_{n=1}^N \frac{w_n^2 \|g_n\|_*^2}{\sum_{m=1}^n w_m} \\
&\leq \hat{\sigma} d^{R_M}(x^*||x_M) \sum_{n=1}^{M-1} w_n + \frac{1}{2\hat{\sigma}} \sum_{n=1}^N \frac{w_n^2 \|g_n\|_*^2}{\sum_{m=1}^n w_m}
\end{aligned}$$

■

Proof of Proposition 5.4.2 Now we use Lemma 5.B.3 to prove the final result. It's easy to see that if g_n is an unbiased stochastic estimate of $\nabla f_n(x_n)$ in Lemma 5.B.3, then the performance bound would hold in expectation since x_n does not depend on g_n. Finally, by

definition of ϵ_{class}, this concludes the proof.

5.C Proof of Section 5.5

5.C.1 Proof of Theorem 5.5.1

Let $w_n = n^d$. The proof is similar to the proof of Proposition 5.4.2 but with weighted cost. First we use Lemma 5.2.1 and bound the series of weighted accumulated loss

$$\mathbb{E}\left[\sum_{n=N_m}^{N_M} w_n J(\pi_n)\right] - \left(\sum_{n=N_m}^{N_M} w_n\right) J(\pi^\star) \leq \frac{C_{\pi^\star}}{1-\gamma} \sum_{n=N_m}^{N_M} w_n l_n(\pi_n)$$

Then we bound the right-hand side by using Lemma 5.B.3,

$$\begin{aligned}\sum_{n=N_m}^{N_M} w_n l_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=N_m}^{N_M} w_n l_n(\pi) &\leq \hat{\sigma} d^{\mathcal{R}} \sum_{n=1}^{N_m-1} w_n + \frac{1}{2\hat{\sigma}} \sum_{n=N_m}^{N_M} \frac{w_n^2 \|g_n\|_*^2}{\sum_{m=1}^n w_m} \\ &\leq \frac{\hat{\sigma} d^{\mathcal{R}} N_m^{d+1}}{d+1} + \frac{d+1}{2\hat{\sigma}} \sum_{n=N_m}^{N_M} \|g_n\|_*^2 n^{d-1}\end{aligned}$$

where we use the fact that $d \geq 0$,

$$\frac{n^{d+1} - (m-1)^{d+1}}{d+1} \leq \sum_{k=m}^{n} k^d \leq \frac{(n+1)^{d+1} - m^{d+1}}{d+1}$$

which implies $\frac{w_n^2}{\sum_{m=1}^n w_m} \leq \frac{(d+1)n^{2d}}{n^{d+1}} \leq (d+1)n^{d-1}$. Combining these two steps, we see that the weighted accumulated loss on average can be bounded by

$$\begin{aligned}&\mathbb{E}\left[\frac{\sum_{n=N_m}^{N_M} w_n J(\pi_n)}{\sum_{n=N_m}^{N_M} w_n}\right] \\ &\leq J(\pi^\star) + \frac{C_{\pi^\star}}{1-\gamma}\left(\epsilon_{\text{class}}^w + \frac{\hat{\sigma} d^{\mathcal{R}} N_m^{d+1}}{(d+1)\sum_{n=N_m}^{N_M} w_n} + \frac{d+1}{2\hat{\sigma}\sum_{n=N_m}^{N_M} w_n} \sum_{n=N_m}^{N_M} \|g_n\|_*^2 n^{d-1}\right)\end{aligned}$$

Because $N_M \geq 2N_m$ and $\frac{x}{1-x} \leq 2x$ for $x \leq \frac{1}{2}$, we have

$$\begin{aligned}
&\frac{N_m^{d+1}}{(d+1)\sum_{n=N_m}^{N_M} w_n} \leq \frac{N_m^{d+1}}{N_M^{d+1} - (N_m - 1)^{d+1}} \\
&\leq \frac{N_m^{d+1}}{N_M^{d+1} - N_m^{d+1}} = \frac{1}{\left(\frac{N_M}{N_m}\right)^{d+1} - 1} \leq 2\left(\frac{N_m}{N_M}\right)^{d+1} \leq 2^{-d}
\end{aligned}$$

and, for $d \geq 1$,

$$\begin{aligned}
\frac{d+1}{\sum_{n=N_m}^{N_M} w_n} \sum_{n=N_m}^{N_M} n^{d-1} &\leq \frac{d+1}{N_M^{d+1} - (N_m - 1)^{d+1}} \frac{d+1}{d}\left((N_M+1)^d - N_m^d\right) \\
&\leq \frac{(d+1)^2}{d} \frac{(N_M+1)^d}{N_M^{d+1} - N_m^{d+1}} \\
&\leq \frac{(d+1)^2}{d} \frac{\frac{1}{N_M}(1+\frac{1}{N_M})^d}{1 - \left(\frac{N_m}{N_M}\right)^{d+1}} \\
&\leq \frac{16d}{3N_M}\left(1 + \frac{1}{N_M}\right)^d \qquad (N_M \geq 2N_m \text{ and } d \geq 1) \\
&\leq \frac{16d}{3N_M} \exp\left(\frac{d}{N_M}\right)
\end{aligned}$$

and for $d = 0$,

$$\frac{d+1}{\sum_{n=N_m}^{N_M} w_n} \sum_{n=N_m}^{N_M} n^{d-1} = \frac{1}{\sum_{n=N_m}^{N_M} 1} \sum_{n=N_m}^{N_M} \frac{1}{n} \leq \frac{\log(N_M) + 1}{N_M - N_m} \leq \frac{2\left(\log(N_M) + 1\right)}{N_M}$$

Thus, by the assumption that $\|g_n\|_* \leq G$ almost surely, the weighted accumulated loss on average has an upper bound

$$\mathbb{E}\left[\frac{\sum_{n=N_m}^{N_M} w_n J(\pi_n)}{\sum_{n=N_m}^{N_M} w_n}\right] \leq J(\pi^\star) + \frac{C_{\pi^\star}}{1-\gamma}\left(\epsilon_{\text{class}}^w + 2^{-d}\hat{\sigma} d^{\mathcal{R}} + \frac{G^2 C_{N_M}/\hat{\sigma}}{N_M}\right)$$

By sampling K according to w_s, this bound directly translates into the the bound on $J(\pi_K)$.

5.C.2 Proof of Theorem 5.6.1

For simplicity, we prove the result of deterministic problems. For stochastic problems, the result can be extended to expected performance, similar to the proof of Proposition 5.4.2. We first define the online learning problem of applying $g_n = \nabla_\theta l_n^\lambda(\pi)|_{\pi=\pi_n}$ to update the policy. In the nth iteration, we define the per-round cost as

$$l_n^\lambda(\pi) = \mathbb{E}_{d^{\pi_n}} \mathbb{E}_\pi[(1-\lambda)A^{\pi_n} + \lambda A^{\pi^\star}] \tag{5.17}$$

With the strongly convexity assumption and large enough step size, similar to the proof for Proposition 5.4.2, we can show that

$$\begin{aligned}\sum_{n=1}^N l_n^\lambda(\pi_n) &\leq \min_{\pi\in\Pi} \sum_{n=1}^N (l_n^\lambda(\pi) + \epsilon_{\text{regret}}^\lambda) \\ &= \min_{\pi\in\Pi} \sum_{n=1}^N \mathbb{E}_{d^{\pi_n}} \mathbb{E}_\pi[(1-\lambda)A^{\pi_n} + \lambda A^{\pi^\star}] + N\epsilon_{\text{regret}}^\lambda\end{aligned}$$

where $\epsilon_{\text{regret}}^\lambda = \tilde{O}\left(\frac{1}{T}\right)$. Note by definition of A^{π_n}, the left-hand-side in the above bound can be written as

$$\frac{1}{N}\sum_{n=1}^N l_n^\lambda(\pi_n) = \sum_{n=1}^N \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi_n}[(1-\lambda)A^{\pi_n} + \lambda A^{\pi^\star}] = \sum_{n=1}^N \lambda \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi_n}[A^{\pi^\star}] \tag{5.18}$$

To relate this to the performance bound, we invoke Lemma 5.2.1 and write

$$
\begin{aligned}
&\sum_{n=1}^{N} J(\pi_n) - ((1-\lambda)J_n^* + \lambda J(\pi^\star)) \\
&= \sum_{n=1}^{N} (1-\lambda)\left(J(\pi_n) - J_n^*\right) + \frac{1}{1-\gamma}\sum_{n=1}^{N} \lambda \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi_n}[A^{\pi^\star}] \\
&\leq \sum_{n=1}^{N} (1-\lambda)\left(J(\pi_n) - J_n^*\right) + \min_{\pi \in \Pi} \frac{1}{1-\gamma}\sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi}[(1-\lambda)A^{\pi_n} + \lambda A^{\pi^\star}] + \frac{N}{1-\gamma}\epsilon_{\text{regret}}^{\lambda} \\
&= \min_{\pi \in \Pi} \frac{1}{1-\gamma}\sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi}[(1-\lambda)Q^{\pi_n} + \lambda A^{\pi^\star}] + \frac{N}{1-\gamma}\epsilon_{\text{regret}}^{\lambda} + \sum_{n=1}^{N} \lambda \left(-\frac{\mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi_n}[Q^{\pi_n}]}{1-\gamma} + J(\pi_n) - J_n^* \right) \\
&= \min_{\pi \in \Pi} \frac{1}{1-\gamma}\sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi}[(1-\lambda)(Q^{\pi_n} - V_{\pi_n}^*) + \lambda(Q^{\pi^\star} - V^{\pi^\star})] + \frac{N}{1-\gamma}\epsilon_{\text{regret}}^{\lambda} \\
&= \min_{\pi \in \Pi} \frac{1}{1-\gamma}\sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi}[(1-\lambda)Q^{\pi_n} + \lambda Q^{\pi^\star}] - \frac{1}{1-\gamma}\sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}} \left[(1-\lambda)V_{\pi_n}^* + \lambda V^{\pi^\star}\right] + \frac{N}{1-\gamma}\epsilon_{\text{regret}}^{\lambda} \\
&\leq \frac{N}{1-\gamma}\epsilon_{\text{class}}^{\lambda} + \frac{N}{1-\gamma}\epsilon_{\text{regret}}^{\lambda}
\end{aligned}
$$

where the second to the last equality is since $\mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi_n}[Q^{\pi_n}] = (1-\gamma)J(\pi_n)$. This concludes the proof.

CHAPTER 6

CONVERGENCE OF VALUE AGGREGATION FOR IMITATION LEARNING

6.1 Introduction

We have shown in the previous chapters that imitation learning (IL) can be used to exploit the domain knowledge about a problem to achieve faster policy learning. In IL, instead of learning a policy from scratch, one leverages a black-box policy $\pi^\star$, called the *expert*, from which the learner can query demonstrations. The goal of IL is to identify a policy π such that its performance is similar to or better than $\pi^\star$. In particular, one powerful approach to IL is based on the idea of data aggregation and online learning (Ross, Gordon, and Bagnell, 2011; Sun et al., 2017) (cf. Chapter 3). The algorithm starts with an empty dataset and an initial policy π_1; in the nth iteration, the algorithm uses the current policy π_n to gather new training data into the current dataset and then a supervised learning problem is solved on the updated dataset to compute the next policy π_{n+1}. By interleaving the optimization and the data collection processes in an online fashion, it can overcome the covariate shift problem in traditional batch IL (Ross, Gordon, and Bagnell, 2011).

This family of algorithms can be realized under the general framework of value aggregation (Ross and Bagnell, 2014), which has gained increasing attention due to its non-asymptotic performance guarantee. After N iterations, a good policy π *exists* in the generated policy sequence $\{\pi_n\}_{n=1}^N$ with performance $J(\pi) \leq J(\pi^\star) + T\epsilon + \tilde{O}(\frac{1}{N})$, where J is the performance index, ϵ is the error due to the limited expressiveness of the policy class, and T is the horizon of the problem. While this result seems strong at first glance, its guarantee concerns only the existence of a good policy and, therefore, is not ideal for stochastic problems. In other words, in order to find the best policy in $\{\pi_n\}_{n=1}^N$ without incurring large statistical error, a sufficient amount of data must be acquired in each iteration,

or all policies have to be memorized for a final evaluation with another large dataset (Ross, Gordon, and Bagnell, 2011). One can also opt to stop the algorithms randomly, as we did in Chapter 5, but that would add extra randomness into the learning processes; in addition, this scheme would rely on *a priori* guarantees, instead of *a posteriori* ones, because the stopping point is chosen independently of how the policy actually learns.

This inconvenience incentivizes practitioners to just return the last policy π_N (Laskey et al., 2017), and, anecdotally, the last policy π_N has been reported to have good empirical performance (Pan et al., 2017a; Ross et al., 2013). Supporting this heuristic is the insight that the last policy π_N is trained with *all* observations and therefore *ideally* should perform the best. Indeed, such idealism works when all the data are sampled i.i.d., as in the traditional batch learning problems (Vapnik, 1998). However, because here new data are collected using the updated policy in each iteration, whether such belief applies depends on the convergence of the distributions generated by the policy sequence.

While Ross and Bagnell (2014) alluded that "...the distribution of visited states converges over the iterations of learning.", we show this is *not* always true—the convergence is rather problem-dependent. In this chapter, we identify a critical stability constant θ that determines the convergence of the policy sequence. We show that there is a simple example (in Section 6.4) in which the policy sequence diverges when $\theta > 1$. In Section 6.5, we provide tight non-asymptotic bounds on the performance of the last policy π_N, in both deterministic and stochastic problems, which implies that the policy sequence always converges when $\theta < 1$. Our new insight also suggests that the stability of the last policy π_N can be recovered by regularization, as discussed in Section 6.6. This chapter is partly based on our paper published as (Cheng and Boots, 2018).

6.2 Problem Setup

We consider solving a discrete-time RL problem. Let $\mathcal{S}$ be the state space and $\mathcal{A}$ be the action space of an agent, where $\mathcal{S}$ embeds time information. Let Π be the class of policies

and let T be the length of the planning horizon.[1] The objective of the agent is to search for a policy $\pi \in \Pi$ to minimize an accumulated cost $J(\pi)$:

$$\min_{\pi \in \Pi} J(\pi) \coloneqq \min_{\pi \in \Pi} \mathbb{E}_{s_0, a_0, \ldots, a_{T-1} \sim \rho^\pi(p)} \left[\sum_{t=0}^{T-1} c(s_t, a_t) \right] \tag{6.1}$$

in which $c(s_t, a_t)$ is the instantaneous cost at time t, and $\rho^\pi(p)$ denotes the trajectory distribution under policy $a_t \sim \pi(a_t|s_t)$ given an initial distribution $p(s_0)$. Note that we do not place assumptions on the structure of $\mathcal{S}$ and $\mathcal{A}$ and the policy class Π, except that we will impose time information into the definition of $\mathcal{S}$ for writing compactness. Note that we write $\mathbb{E}_{a \sim \pi|s}$ even if the policy is deterministic.

We denote $Q^\pi(s, a)$ as the Q-function under policy π and $V^\pi(s) = \mathbb{E}_{a \sim \pi|s}[Q^\pi(s, a)]$ as the associated value function. In addition, we introduce some shorthand: we denote $d_t^\pi(s)$ as the state distribution at time t generated by running the policy π for the first t steps, and define the average distribution $d^\pi(s) = \frac{1}{T} \sum_{t=0}^{T-1} d_t^\pi(s)$. Due to space limitations, we will often omit explicit dependencies on random variables in expectations, e.g. we will write $\min_{\pi \in \Pi} \mathbb{E}_{s \sim d^\pi} \mathbb{E}_{a \sim \pi|s} [c(s, a)]$ also as

$$\min_{\pi \in \Pi} \mathbb{E}_{d^\pi} \mathbb{E}_\pi [c] , \tag{6.2}$$

which is equivalent to $\min_{\pi \in \Pi} \frac{1}{T} J(\pi)$ (by definition of d^π).

6.3 Value Aggregation

Solving general RL problems is challenging. In this chapter, we focus on a particular scenario, in which the agent, or the *learner*, has access to an *expert* policy $\pi^\star$ from which the learner can query demonstrations. Here we embrace a general notion of expert. While it is often preferred that the expert is nearly optimal in (6.1), the expert here can be *any* policy, e.g. the agent's initial policy. Note, additionally, that the RL problem considered

[1] A similar analysis can be applied to discounted infinite-horizon problems.

here is not necessarily directly related to a real-world application; it can be a surrogate problem which arises in solving the true problem.

The goal of IL is to find a policy π that outperforms or behaves similarly to the expert $\pi^\star$ in the sense that $J(\pi) \leq J(\pi^\star) + O(T)$. That is, we treat IL as performing a robust, approximate policy iteration step from $\pi^\star$: ideally IL should lead to a policy that outperforms the expert, but it at least returns a policy that performs similarly to the expert.

AGGREVATTE (Aggregate Value to Imitate) is an IL algorithm proposed by Ross and Bagnell (2014) based on the idea of online learning (Hazan, 2016). Here we give a compact derivation and discuss its important features in preparation for the analysis in Section 6.5. To this end, we recall the performance difference lemma due to Kakade and Langford (2002), which will be used as the foundation to derive AGGREVATTE (cf. Chapter 2).

Lemma 6.3.1. *Let π and π' be two policies and $A^{\pi'}(s,a) = Q^{\pi'}(s,a) - V^{\pi'}(s)$ be the (dis)advantage function with respect to running π'. Then it holds that*

$$J(\pi) = J(\pi') + T\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}[A^{\pi'}(s,a)]. \tag{6.3}$$

6.3.1 Motivation

The main idea of AGGREVATTE is to minimize the performance difference between the learner's policy and the expert policy, which, by Lemma 6.3.1, is given as $\frac{1}{T}\left(J(\pi) - J(\pi^\star)\right) = \mathbb{E}_{d^\pi}\mathbb{E}_\pi[A^{\pi^\star}]$. AGGREVATTE can be viewed as solving an RL problem with $A^{\pi^\star}$ as the instantaneous cost:

$$\min_{\pi\in\Pi} \mathbb{E}_{d^\pi}\mathbb{E}_\pi\left[A^{\pi^\star}\right]. \tag{6.4}$$

Although the transformation from (6.2) to (6.4) seems trivial (it is just a constant shift of the objective function), it unveils some critical properties. Most importantly, the range of the problem in (6.4) is normalized. For example, regardless of the original definition

of c, if $\Pi \ni \pi^\star$, there exists at least a policy $\pi \in \Pi$ such that (6.4) is non-positive (i.e. $J(\pi) \leq J(\pi^\star)$). As now the problem (6.4) is relative, it becomes possible to place a qualitative assumption to bound the performance in (6.4) in terms of some measure of expressiveness of the policy class Π.

We formalize this idea into Assumption 6.3.1, which is one of the core assumptions implicitly imposed by Ross and Bagnell (2014).[2] To simplify the notation, we define a bifunction F such that for any two policies π, π'

$$F(\pi, \pi') \coloneqq \mathbb{E}_{d^\pi} \mathbb{E}_{\pi'} \left[A^{\pi^\star} \right] \tag{6.5}$$

This function captures the main structure in (6.4). By separating the roles of π (which controls the state distribution) and π' (which controls the reaction/prediction), the performance of a policy class Π relative to an expert $\pi^\star$ can be characterized with the approximation error in a supervised learning problem.

Assumption 6.3.1. Given a policy $\pi^\star$, the policy class Π satisfies that for arbitrary sequence of policies $\{\pi_n \in \Pi\}_{n=1}^N$, there exists a small constant $\epsilon_{\Pi,\pi^\star}$ such that

$$\min_{\pi \in \Pi} \frac{1}{N} l_{1:N}(\pi) \leq \epsilon_{\Pi,\pi^\star}, \tag{6.6}$$

where $l_n(\pi) \coloneqq F(\pi_n, \pi)$ and $l_{1:n}(\pi) = \sum_{n=1}^N l_n(\pi)$.

This assumption says that there exists at least a policy $\pi \in \Pi$ which is as good as $\pi^\star$ in the sense that π can predict $\pi^\star$ well in a cost-sensitive supervised learning problem, with small error $\epsilon_{\Pi,\pi^\star}$, under the average state distribution generated by an *arbitrary* policy sequence $\{\pi_n \in \Pi\}_{n=1}^N$.

Following this assumption, AGGREVATTE exploits another critical structural property of the problem.

[2]The assumption is implicitly made when Ross and Bagnell (2014) assume the existence of ϵ_{class} in Theorem 2.1 on page 4.

Assumption 6.3.2. $\forall \pi \in \Pi$, $F(\pi, \pi')$ is a strongly convex function in π'.

While Ross and Bagnell (2014) did not explicitly discuss under which condition Assumption 6.3.2 holds, here we point out some examples (proved in Section 6.A).

Proposition 6.3.1. *Suppose* Π *consists of deterministic linear policies (i.e.* $a = \phi(s)^\top x$ *for some feature map* $\phi(s)$ *and weight* x*) and* $\forall s \in \mathcal{S}$, $c(s, \cdot)$ *is strongly convex. Assumption 6.3.2 holds under any of the following when the state distribution is diverse enough:*

1. $V^{\pi^\star}(s)$ *is constant over* $\mathcal{S}$ *(in this case* $A^{\pi^\star}(s, a)$ *is equivalent to* $c(s, a)$ *up to a constant in* a*)*
2. *The problem is continuous-time and the dynamics are affine in action.*

We further note that AGGREVATTE has demonstrated impressive empirical success even when Assumption 6.3.2 cannot be verified (Pan et al., 2017a; Sun et al., 2017).

6.3.2 Algorithm and Performance

Given Assumption 6.3.2, AGGREVATTE treats $l_n(\cdot)$ as the per-round loss in an online convex optimization problem and updates the policy sequence as follows: Let π_1 be an initial policy. In the nth iteration of AggreVaTe, the policy is updated by[3]

$$\pi_{n+1} = \arg\min_{\pi \in \Pi} l_{1:n}(\pi). \tag{6.7}$$

After N iterations, the best policy in the sequence $\{\pi_n\}_{n=1}^N$ is returned, i.e. $\pi = \hat{\pi}_N$, where

$$\hat{\pi}_N \coloneqq \arg\min_{\pi \in \{\pi_n\}_{n=1}^N} J(\pi). \tag{6.8}$$

[3]We adopt a different notation from (Ross and Bagnell, 2014), in which the per-round loss $\mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi} \left[Q^{\pi^\star} \right]$ was used. Note these two terms are equivalent up to an additive constant, as the optimization here is over π with π_n fixed.

As the update rule (6.7) (aka Follow-the-Leader) has a sublinear regret, it can be shown that (cf. Section 6.5.1)

$$J(\hat{\pi}_N) \leq J(\pi^\star) + T\left(\epsilon_{\text{class}} + \epsilon_{\text{regret}}\right), \tag{6.9}$$

in which $\epsilon_{\text{regret}} = \tilde{O}(\frac{1}{N})$ is the average regret and

$$\epsilon_{\text{class}} \coloneqq \min_{\pi \in \Pi} \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}} \left[\mathbb{E}_{\pi}[Q^{\pi^\star}] - \mathbb{E}_{\pi^\star}[Q^{\pi^\star}]\right]$$

compares the best policy in the policy class Π and the expert policy $\pi^\star$. The term ϵ_{class} can be negative if there exists a policy in Π that is better than $\pi^\star$ under the average state distribution, $\frac{1}{N}\sum_{n=1}^{N} d^{\pi_n}$, generated by AGGREVATTE. By Assumption 6.3.1, $\epsilon_{\text{class}} \leq \epsilon_{\Pi,\pi^\star}$; we know ϵ_{class} at least should be small.

The performance bound in (6.9) satisfies the requirement of IL that $J(\hat{\pi}_N) \leq J(\pi^\star) + O(T)$. Especially because ϵ_{class} can be non-positive, AGGREVATTE can be viewed as robustly performing one approximate policy iteration step from $\pi^\star$.

One notable special case of AGGREVATTE is DAGGER (Ross, Gordon, and Bagnell, 2011). DAGGER tackles the problem of solving an unknown RL problem by imitating a desired policy $\pi^\star$. The reduction to AGGREVATTE can be seen by setting $c(s,a) = \mathbb{E}_{a^* \sim \pi^\star}[\|a - a^*\|]$ in (6.1). In this case, $\pi^\star$ is optimal for this specific choice of cost and therefore $V^{\pi^\star}(s) = 0$. By Proposition 6.3.1, $A^{\pi^\star}(s,a) = c(s,a)$ and ϵ_{class} reduces to $\min_{\pi\in\Pi} \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}}\mathbb{E}_{\pi}[c] \geq 0$, which is related to the expressiveness of the policy class.

6.4 Guarantee On the Last Policy?

The performance bound in Section 6.3 implicitly assumes that the problem is either deterministic or that infinite samples are available in each iteration. For stochastic problems, $l_{1:n}$ can be approximated by finite samples or by function approximators (Ross and Bagnell,

2014). Suppose m samples are collected in each iteration to approximate l_n. An additional error in $O(\frac{1}{\sqrt{mN}})$ will be added to the performance of $\hat{\pi}_N$. However, in practice, another constant statistical error[4] in $O(\frac{1}{m})$ is introduced when one attempts to identify $\hat{\pi}_N$ from the sequence $\{\pi_n\}_{n=1}^N$.

This practical issue motivates us to ask whether a similar guarantee applies to the last policy π_N so that the selection process to find $\hat{\pi}_N$ can be removed. In fact, the last policy π_n has been reported to have good performance empirically (Pan et al., 2017a; Ross et al., 2013). It becomes interesting to know what one can say about π_N. It turns out that running AGGREVATTE does not always yield a policy sequence $\{\pi_n\}$ with reasonable performance, as given in the example below.

A Motivating Example Consider a two-stage deterministic optimal control problem:

$$\min_{\pi\in\Pi} J(\pi) = \min_{\pi\in\Pi} c_1(s_1, a_1) + c_2(s_2, a_2) \tag{6.10}$$

where the transition and costs are given as

$$s_1 = 0, \quad s_2 = \theta(s_1 + a_1),$$
$$c_1(s_1, a_1) = 0, \quad c_2(s_2, a_2) = (s_2 - a_2)^2.$$

Since the problem is deterministic, we consider a policy class Π consisting of open-loop stationary deterministic policies, i.e. $a_1 = a_2 = x$ for some x (for convenience π and x will be used interchangeably). It can be easily seen that Π contains a globally optimal policy, namely $x = 0$. We perform AGGREVATTE with a feedback expert policy $a_t^* = s_t$ and some initial policy $|x_1| > 0$. While it is a custom to initialize $x_1 = \arg\min_{x\in\mathcal{X}} F(x^*, x)$ (which in this case would ideally return $x_1 = 0$), setting $|x_1| > 0$ simulates the effect of

[4]The original analysis in the stochastic case by Ross and Bagnell (2014) only guarantees the existence of a good policy in the sequence. The $O(\frac{1}{m})$ error is due to identifying the best policy (Lee, Bartlett, and Williamson, 1998) (as the function is strongly convex) and the $O(\frac{1}{\sqrt{mN}})$ error is the generalization error (Cesa-Bianchi, Conconi, and Gentile, 2004).

finite numerical precision.

We consider two cases ($\theta > 1$ or $\theta < 1$) to understand the behavior of AGGREVATTE. First, suppose $\theta > 1$. Without loss generality, take $\theta = 10$ and $x_1 = 1$. We can see running AGGREVATTE will generate a divergent sequence $x_2 = 10, x_3 = 55, x_4 = 220 \ldots$ (in this case AGGREVATTE would return x_1 as the best policy). Since $J(x) = (\theta - 1)^2 x^2$, the performance $\{J(x_n)\}$ is an increasing sequence. Therefore, we see even in this simple case, which can be trivially solved by gradient descent in $O(\frac{1}{n})$, using AGGREVATTE results in a sequence of policies with degrading performance, though the policy class Π includes a globally optimal policy. Now suppose on the contrary $\theta < 1$. We can see that $\{x_n\}$ asymptotically converges to $x^* = 0$.

This example illustrates several important properties of AGGREVATTE. It shows that whether AGGREVATTE can generate a reasonable policy sequence or not depends on intrinsic properties of the problem (i.e. the value of θ). The non-monotonic property was also empirically found in Laskey et al. (2017). In addition, it shows that $\epsilon_{\Pi,\pi^\star}$ can be large while Π contains an optimal policy.[5] This suggests that Assumption 6.3.1 may be too strong, especially in the case where Π does not contain $\pi^\star$.

6.5 Theoretical Analysis

Motivated by the example in Section 6.4, we investigate the convergence of the policy sequence generated by AGGREVATTE in general problems. We assume the policy class Π consists of policies parametrized by some parameter $x \in \mathcal{X}$, in which $\mathcal{X}$ is a convex set in a normed space with norm $\|\cdot\|$ (and $\|\cdot\|_*$ as its dual norm). With abuse of notation, we abstract the RL problem in (6.4) as

$$\min_{x \in \mathcal{X}} F(x, x) \tag{6.11}$$

[5] In this example, $\epsilon_{\Pi,\pi^\star}$ can be arbitrarily large unless $\mathcal{X}$ is bounded. However, even when $\epsilon_{\Pi,\pi^\star}$ is bounded, the performance of the policy sequence can be non-monotonic.

where we overload the notation $F(\pi, \pi')$ defined in (6.5) as $F(\pi, \pi') = F(x, y)$ when $\pi, \pi' \in \Pi$ are parametrized by $x, y \in \mathcal{X}$, respectively. Similarly, we will write $l_n(x) = F(x_n, x)$ for short. In this new notation, AGGREVATTE's update rule in (6.7) can be simply written as $x_{n+1} = \arg\min_{x \in \mathcal{X}} l_{1:n}(x)$.

Here we will focus on the bound on $F(x, x)$, because, for π parameterized by x, this result can be directly translated to a bound on $J(\pi)$: by definition of F in (6.5) and Lemma 6.3.1, $J(\pi) = J(\pi^\star) + TF(\pi, \pi)$. For simplicity, we will assume for now F is deterministic; the convergence in stochastic problems will be discussed at the end of the section.

6.5.1 Classical Result

For completeness, we restate the structural assumptions made by AGGREVATTE in terms of $\mathcal{X}$ and review the known convergence of AGGREVATTE (Ross and Bagnell, 2014).

Assumption 6.5.1. Let ∇_2 denote the derivative with respect to the second argument.

1. F is uniformly α-strongly convex in the second argument: $\forall x, y, z \in \mathcal{X}$, $F(z, x) \geq F(z, y) + \langle \nabla_2 F(z, y), x - y \rangle + \frac{\alpha}{2}\|x - y\|^2$.

2. F is uniformly G_2-Lipschitz continuous in the second argument: $\forall x, y, z \in \mathcal{X}$, $|F(z, x) - F(z, y)| \leq G_2 \|x - y\|$.

Assumption 6.5.2. $\forall \{x_n \in \mathcal{X}\}_{n=1}^N$, there exists a small constant $\epsilon_{\Pi,\pi^\star}$ such that $\min_{x \in \mathcal{X}} \frac{1}{N} l_{1:N}(x) \leq \epsilon_{\Pi,\pi^\star}$.

Theorem 6.5.1. *Under Assumption 6.5.1 and 6.5.2,* AGGREVATTE *generates a sequence such that, for all* $N \geq 1$*,*

$$F(\hat{x}_N, \hat{x}_N) \leq \frac{1}{N} \sum_{n=1}^{N} l_n(x_n) \leq \epsilon_{\Pi,\pi^\star} + \frac{G_2^2}{2\alpha} \frac{\ln(N) + 1}{N}$$

where $\hat{x}_N \coloneqq \arg\min_{x \in \{x_n\}_{n=1}^N} F(x, x)$.

Proof. Here we present a sketch (see Section 6.A for details). The first inequality is straightforward. To bound the average performance, it can be shown that $\sum_{n=1}^{N} l_n(x_n) \leq \min_{x \in \mathcal{X}} l_{1:N}(x) + \sum_{n=1}^{N} l_{1:n}(x_n) - l_{1:n}(x_{n+1})$. Since x_n minimizes $l_{1:n-1}$ and $l_{1:n}$ is $n\alpha$-strongly convex , $l_{1:n}(x_n)$ is upper bounded by $l_{1:n-1}(x_n) + \frac{\|\nabla l_n(x_n)\|_*^2}{2\alpha n}$, where $\|\nabla l_n(x_n)\|_* \leq G_2$. This concludes the proof. ■

6.5.2 New Structural Assumptions

AGGREVATTE can be viewed as an attempt to solve the optimization problem in (6.11) without any information (not even continuity) regarding how $F(x, x)$ changes with perturbations in the first argument. Since making even a local improvement for general Lipschitz continuous problems is known to be NP-hard (Nesterov, 2013), the classical performance guarantee of AGGREVATTE is made possible, only because of the additional structure given in Assumption 6.5.2. However, as discussed in Section 6.4, Assumption 6.5.2 can be too strong and is yet insufficient to determine if the performance of the last policy can improve over iterations. Therefore, to analyze the performance of the last policy, we require additional structure on F.

Here we introduce a continuity assumption.

Assumption 6.5.3. $\nabla_2 F$ is uniformly β-Lipschitz continuous in the first argument: $\forall x, y, z \in \mathcal{X}$ $\|\nabla_2 F(x, z) - \nabla_2 F(y, z)\|_* \leq \beta \|x - y\|$.

Because the first argument of F in (6.5) defines the change of state distribution, Assumption 6.5.3 basically requires that the expectation over d^π changes continuously with respect to π, which is satisfied in most RL problems. Intuitively, this quantifies the difficulty of a problem in terms of how sensitive the state distribution is to policy changes.

In addition, we relax Assumption 6.5.2. As shown in Section 6.4, Assumption 6.5.2 is sometimes too strong, because it might not be satisfied even when Π contains a globally optimal policy. In the analysis of convergence, we instead rely on a necessary condition of Assumption 6.5.2, which is satisfied by the example in Section 6.4.

Assumption 6.5.4. Let π be a policy parametrized by x. There exists a small constant $\tilde{\epsilon}_{\pi,\pi^\star}$ such that $\forall x \in \mathcal{X}$, $\min_{y\in\mathcal{X}} F(x,y) \leq \tilde{\epsilon}_{\Pi,\pi^\star}$.

Compared with the global Assumption 6.5.2, the relaxed condition here is only *local*: it only requires the existence of a good policy with respect to the state distribution visited by running a *single* policy. It can be easily shown that $\tilde{\epsilon}_{\Pi,\pi^\star} \leq \epsilon_{\Pi,\pi^\star}$.

6.5.3 Guarantee on the Last Policy

In our analysis, we define a stability constant $\theta = \frac{\beta}{\alpha}$. One can verify that this definition agrees with the θ used in the example in Section 6.4. This stability constant will play a crucial role in determining the convergence of $\{x_n\}$, similar to the spectral norm of the Jacobian matrix in discrete-time dynamical systems (Antsaklis and Michel, 2007). We have already shown above that if $\theta > 1$ there is a problem such that AGGREVATTE generates a divergent sequence $\{x_n\}$ with degrading performance over iterations. We now show that if $\theta < 1$, then $\lim_{n\to\infty} F(x_n, x_n) \leq \tilde{\epsilon}_{\Pi,\pi^\star}$ and moreover $\{x_n\}$ is convergent.

Theorem 6.5.2. *Suppose Assumptions 6.5.1, 6.5.3, and 6.5.4 are satisfied. Let $\theta = \frac{\beta}{\alpha}$. Then for all $N \geq 1$ it holds*

$$F(x_N, x_N) \leq \tilde{\epsilon}_{\Pi,\pi^\star} + \frac{\left(\theta e^{1-\theta} G_2\right)^2}{2\alpha} N^{2(\theta-1)}$$

and $\|x_N - \bar{x}_N\| = \frac{G_2 e^{1-\theta}}{\alpha} N^{\theta-1}$, where $\bar{x}_N = \frac{1}{N} x_{1:N}$. In particular, if $\theta < 1$, then $\{x_n\}_{n=1}^{\infty}$ is convergent

Theorem 6.5.2 implies that the stability and convergence of AGGREVATTE depends solely on the problem properties. If the state distribution d^π is sensitive to minor policy changes, running AGGREVATTE would fail to provide any guarantee on the last policy. Moreover, Theorem 6.5.2 also characterizes the performance of the average policy $\bar{x}_N$ when $\theta < 1$, .

The upper bound in Theorem 6.5.2 is tight, as indicated in the next theorem. Note a lower bound on $F(x_N, x_N)$ leads directly to a lower bound on $J(\pi_N)$ for π_N parametrized by x_N.

Theorem 6.5.3. *There is a problem such that running* AGGREVATTE *for N iterations results in $F(x_N, x_N) \geq \tilde{\epsilon}_{\Pi,\pi^\star} + \Omega(N^{2(\theta-1)})$. In particular, if $\theta > 1$, the policy sequence and performance sequence diverge.*

Proof. The proof is based on analyzing the sequence in the example in Section 6.4. See Section 6.A. ■

6.5.4 Proof of Theorem 6.5.2

Now we give the proof of Theorem 6.5.2. Without using the first-order information of F in the first argument, we construct our analysis based on the convergence of an intermediate quantity, which indicates how fast the sequence concentrates toward its last element:

$$S_n := \frac{\sum_{k=1}^{n-1} \|x_n - x_k\|}{n-1} \tag{6.12}$$

which is defined $n \geq 2$ and $S_2 = \|x_2 - x_1\|$.

First, we use Assumption 6.5.3 to strengthen the bound $\|x_{n+1} - x_n\| = O(\frac{1}{n})$ used in Theorem 6.5.1 by techniques from online learning with prediction (Rakhlin and Sridharan, 2012).

Lemma 6.5.1. *Under Assumptions 6.5.1 and 6.5.3, running* AGGREVATTE *gives, for $n \geq 2$, $\|x_{n+1} - x_n\| \leq \frac{\theta S_n}{n}$.*

Proof. First, because $l_{1:n}$ is $n\alpha$-strongly convex,

$$\begin{aligned}\frac{n\alpha}{2}\|x_{n+1} - x_n\|^2 &\leq l_{1:n}(x_n) - l_{1:n}(x_{n+1}) \\ &\leq \langle \nabla l_{1:n}(x_n), x_n - x_{n+1}\rangle - \frac{n\alpha}{2}\|x_n - x_{n+1}\|^2.\end{aligned}$$

Let $\bar{l}_n = \frac{1}{n} l_{1:n}$. The above inequality implies

$$\begin{aligned} n\alpha \|x_{n+1} - x_n\|^2 &\leq \langle \nabla l_n(x_n), x_n - x_{n+1} \rangle \\ &\leq \left\langle \nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n), x_n - x_{n+1} \right\rangle \\ &\leq \|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_* \|x_n - x_{n+1}\| \\ &\leq \beta S_n \|x_n - x_{n+1}\| \end{aligned}$$

where the second inequality is due to $x_n = \arg\min_{x \in \mathcal{X}} l_{1:n-1}(x)$ and the last inequality is due to Assumption 6.5.3. Thus, $\|x_n - x_{n+1}\| \leq \frac{\beta S_n}{\alpha n}$. ■

Using the refined bound provided by Lemma 6.5.1, we can bound the progress of S_n.

Proposition 6.5.1. *Under the assumptions in Lemma 6.5.1, for $n \geq 2$, $S_n \leq e^{1-\theta} n^{\theta-1} S_2$ and $S_2 = \|x_2 - x_1\| \leq \frac{G_2}{\alpha}$.*

Proof. The bound on $S_2 = \|x_2 - x_1\|$ is due to that $x_2 = \arg\min_{x \in \mathcal{X}} l_1(x)$ and that l_1 is α-strongly convex and G_2-Lipschitz continuous.

To bound S_n, first we bound S_{n+1} in terms of S_n by

$$\begin{aligned} S_{n+1} &\leq \left(1 - \frac{1}{n}\right) S_n + \|x_{n+1} - x_n\| \\ &\leq \left(1 - \frac{1}{n} + \frac{\theta}{n}\right) S_n = \left(1 - \frac{1-\theta}{n}\right) S_n \end{aligned}$$

in which the first in equality is due to triangular inequality (i.e. $\|x_k - x_{n+1}\| \leq \|x_k - x_n\| + \|x_n - x_{n+1}\|$) and the second inequality is due to Lemma 6.5.1. Let $P_n = \ln S_n$. Then we can bound $P_n - P_2 \leq \sum_{k=2}^{n-1} \ln\left(1 - \frac{1-\theta}{k}\right) \leq \sum_{k=2}^{n-1} -\frac{1-\theta}{k} \leq -(1-\theta)(\ln n - 1)$, where we use the facts that $\ln(1+x) \leq x$, $\sum_{k=1}^{n} \frac{1}{k} \geq \ln(n+1)$. This implies $S_n = \exp(P_n) \leq e^{1-\theta} n^{\theta-1} S_2$. ■

More generally, define $S_{m:n} = \frac{\sum_{k=m}^{n-1} \|x_n - x_k\|}{n-m}$ (i.e. $S_n = S_{1:n}$). Using Proposition 6.5.1, we give a bound on $S_{m:n}$. We see that the convergence of $S_{m:n}$ depends mostly on n not m.

(The proof is given in appendix of this chapter.)

Corollary 6.5.1. *Under the assumptions in Lemma 6.5.1, for* $n > m$, $S_{m:n} \leq O(\frac{\theta}{(n-m)m^{2-\theta}} + \frac{1}{n^{1-\theta}})$.

Now we are ready prove Theorem 6.5.2 by using the concentration of S_n in Proposition 6.5.1.

Proof of Theorem 6.5.2. First, we prove the bound on $F(x_N, x_N)$. Let $x_n^* := \arg\min_{x\in\mathcal{X}} l_n(x)$ and let $\bar{l}_n = \frac{1}{n} l_{1:n}$. Then by α-strongly convexity of l_n,

$$\begin{aligned} l_n(x_n) - \min_{x\in\mathcal{X}} l_n(x) &\leq \langle \nabla l_n(x_n), x_n - x_n^* \rangle - \frac{\alpha}{2}\|x_n - x_n^*\|^2 \\ &\leq \left\langle \nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n), x_n - x_n^* \right\rangle - \frac{\alpha}{2}\|x_n - x_n^*\|^2 \\ &\leq \|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_* \|x_n - x_n^*\| - \frac{\alpha}{2}\|x_n - x_n^*\|^2 \\ &\leq \frac{\|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_*^2}{2\alpha} \leq \frac{\beta^2}{2\alpha} S_n^2 \end{aligned}$$

where the second inequality uses the fact that $x_n = \arg\min_{x\in\mathcal{X}} \bar{l}_{n-1}(x)$, the second to the last inequality takes the maximum over $\|x_n - x_n^*\|$, and the last inequality uses Assumption 6.5.3. Therefore, to bound $l_n(x_n)$, we can use Proposition 6.5.1 and Assumption 6.5.4:

$$l_n(x_n) \leq \min_{x\in\mathcal{X}} l_n(x) + \frac{\beta^2}{2\alpha} S_n^2 \leq \tilde{\epsilon}_{\Pi,\pi^\star} + \frac{\beta^2}{2\alpha}\left(e^{1-\theta} n^{\theta-1} \frac{G_2}{\alpha}\right)^2$$

Rearranging the terms gives the bound in Theorem 6.5.2, and that $\|x_n - \bar{x}_n\| \leq S_n$ gives the second result.

Now we show the convergence of $\{x_n\}$ under the condition $\theta < 1$. It is sufficient to show that $\lim_{n\to\infty} \sum_{k=1}^n \|x_k - x_{k+1}\| < \infty$. To see this, we apply Lemma 6.5.1 and Proposition 6.5.1: for $\theta < 1$, $\sum_{k=1}^n \|x_k - x_{k+1}\| \leq \|x_1 - x_2\| + \sum_{k=2}^n \frac{\theta}{k} S_k \leq c_1 + c_2 \sum_{k=2}^n \frac{\theta}{k} \frac{S_2}{k^{1-\theta}} < \infty$, where $c_1, c_2 \in O(1)$. ■

6.5.5 Stochastic Problems

We analyze the convergence of AGGREVATTE in stochastic problems using finite-sample approximation: Define $f(x; s) = \mathbb{E}_\pi[A^{\pi^\star}]$ such that $l_n(x) = \mathbb{E}_{s\sim d^{\pi_n}}[f(x; s)]$. Instead of using $l_n(\cdot)$ as the per-round loss in the nth iteration, we take its finite samples approximation $g_n(\cdot) = \sum_{k=1}^{m_n} f(\cdot; s_{n,k})$, where m_n is the number of independent samples collected in the nth iteration under distribution d^{π_n}. That is, the update rule in (6.7) in stochastic setting is modified to $\pi_{n+1} = \arg\min_{\pi\in\Pi} g_{1:n}(\pi)$.

Theorem 6.5.4. *In addition to Assumptions 6.5.3 and 6.5.4, assume $f(x; s)$ is α-strongly convex in x and $\|\nabla f(x; s)\|_* < G_2$ almost surely. Let $\theta = \frac{\beta}{\alpha}$ and suppose $m_n = m_0 n^r$ for some $r \geq 0$. For all $N > 0$, with probability at least $1 - \delta$,*

$$F(x_N, x_N) \leq \tilde{\epsilon}_{\Pi,\pi^\star} + \tilde{O}\left(\frac{\theta^2}{c}\frac{\ln(1/\delta) + C_\mathcal{X}}{N^{\min\{r,2,2-2\theta\}}}\right) + \tilde{O}\left(\frac{\ln(1/\delta) + C_\mathcal{X}}{cN^{\min\{2,1+r\}}}\right)$$

where $c = \frac{\alpha}{G_2^2 m_0}$ and $C_\mathcal{X}$ is a constant[6] *of the complexity of Π.*

Proof. The proof is similar to the proof of Theorem 6.5.2. To handle the stochasticity, we use a generalization of Azuma-Hoeffding inequality to vector-valued martingales (Hayes, 2005) to derive a high-probability bound on $\|\nabla g_n(x_n) - \nabla l_n(x_n)\|_*$ and a uniform bound on $\sup_{x\in\mathcal{X}} \frac{1}{n}\|\nabla g_{1:n}(x) - \nabla l_{1:n}(x)\|_*$. These error bounds allow us to derive a stochastic version of Lemma 6.5.1, Proposition 6.5.1, and then the performance inequality in the proof of Theorem 6.5.2. See Section 6.B for the complete proof. ■

The growth of sample size m_n over iterations determines the main behavior of AGGREVATTE in stochastic problems. For $r = 0$, compared with Theorem 6.5.2, Theorem 6.5.4 has an additional constant error in $\tilde{O}(\frac{1}{m_0})$, which is comparable to the stochastic error in selecting the best policy in the classical approach. However, the error here is due to approx-

[6]The constant $C_\mathcal{X}$ can be thought as $\ln|\mathcal{X}|$, where $|\mathcal{X}|$ measures the size of $\mathcal{X}$ in e.g. Rademacher complexity or covering number (Mohri, Rostamizadeh, and Talwalkar, 2012). For example, $\ln|\mathcal{X}|$ can be linear in $\dim\mathcal{X}$.

imating the gradient ∇l_n rather than the objective function l_n. For $r > 0$, by slightly taking more samples over iterations (e.g. $r = 2 - 2\theta$), we see the convergence rate can get closer to $\tilde{O}(N^{2-2\theta})$ as in the ideal case given by Theorem 6.5.2. However, it cannot be better than $\tilde{O}(\frac{1}{N})$. Therefore, for stochastic problems, a stability constant $\theta < 1/2$ and a growing rate $r > 1$ does not contribute to faster convergence as opposed to the deterministic case in Theorem 6.5.2.

Note while our analysis here is based on finite-sample approximation $g_n(\cdot) = \sum_{k=1}^{m_n} f(\cdot; s_{n,k})$, the same technique can also be applied to the scenario in the bandit setting and another online regression problem is solved to learn $l_n(\cdot)$ as in the case considered by Ross and Bagnell (2014). A discussion is given in Section 6.C.

The analysis given as Theorem 6.5.4 can be viewed as a generalization of the analysis of Empirical Risk Minimization (ERM) to non-i.i.d. scenarios, where the distribution depends on the decision variable. For optimizing a strongly convex objective function with i.i.d. samples, it has been shown by Shalev-Shwartz et al. (2009) that x_N exhibits a fast convergence to the optimal performance in $O(\frac{1}{N})$. By specializing our general result in Theorem 6.5.4 with $\theta, r = 0$ to recover the classical i.i.d. setting, we arrive at a bound on the performance of x_N in $\tilde{O}(\frac{1}{N})$, which matches the best known result up to a log factor. However, Theorem 6.5.4 is proved by a completely different technique using the martingale concentration of the gradient sequence. In addition, by Theorem 6.5.2, the theoretical results of x_N here can directly translate to that of the mean policy $\bar{x}_N$, which matches the bound for the average decision $\bar{x}_N$ given by Kakade and Tewari (2009).

6.6 Regularization

We have shown that whether AGGREVATTE generates a convergent policy sequence and a last policy with the desired performance depends on the stability constant θ. Here we show that by adding regularization to the problem we can make the problem stable. For simplicity, here we consider deterministic problems or stochastic problems with infinite

samples.

6.6.1 Mixing Policies

We first consider the idea of using mixing policies to collect samples, which was originally proposed as a heuristic by Ross, Gordon, and Bagnell (2011). It works as follows: in the nth iteration of AGGREVATTE, instead of using $F(\pi_n, \cdot)$ as the per-round loss, it uses $\hat{F}(\pi_n, \cdot)$ which is defined by

$$\hat{F}(\pi_n, \pi) = \mathbb{E}_{d^{\tilde{\pi}_n}} \mathbb{E}_{\pi}[A^{\pi^\star}] \tag{6.13}$$

The state distribution $d^{\tilde{\pi}_n}(s)$ is generated by running $\pi^\star$ with probability q and π_n with probability $1-q$ at each time step. Originally, Ross, Gordon, and Bagnell (2011) propose to set q to decay exponentially over the iterations of AGGREVATTE. (The proofs are given in Section 6.A.)

Here we show that the usage of mixing policies also has the effect of stabilizing the problem.

Lemma 6.6.1. *Let $\|p_1 - p_2\|_1$ denote the total variational distance between distributions p_1 and p_2. Assume[7] for any policy π, π' parameterized by x, y it satisfies $\frac{1}{T}\sum_{t=0}^{T-1} \|d_t^{\pi} - d_t^{\pi'}\|_1 \leq \frac{\beta}{2G_2}\|x-y\|$ and assume $\|\nabla_x \mathbb{E}_{\pi}[A^{\pi^\star}](s)\|_* \leq G_2$. Then $\nabla_2 F$ is uniformly $(1-q^T)\beta$-Lipschitz continuous in the second argument.*

By Lemma 6.6.1, if $\theta > 1$, then choosing a fixed $q > (1 - \frac{1}{\theta})^{1/T}$ ensures the stability constant of $\hat{F}$ to be $\hat{\theta} < 1$. However, stabilizing the problem in this way incurs a constant cost as shown in Corollary 6.6.1.

Corollary 6.6.1. *Suppose $\mathbb{E}_{\pi}[A^{\pi^\star}] < M$ for all π. Define $\Delta_N = \frac{(\hat{\theta}e^{1-\hat{\theta}}G_2)^2}{2\alpha} N^{2(\hat{\theta}-1)}$. Then under the assumptions in Lemma 6.6.1 and Assumption 6.5.1.1, running* AGGREVATTE

[7] These two are sufficient to Assumption 6.5.1.2 and 6.5.3.

with $\tilde{F}$ in (6.13) *and a mixing rate q gives*

$$F(x_N, x_N) \leq \Delta_N + \tilde{\epsilon}_{\Pi,\pi^\star} + 2M \min(1, Tq)$$

Proof. The proof is similar to Lemma 6.6.1 and the proof of Ross, Gordon, and Bagnell, 2011, Theorem 4.1. ■

6.6.2 Weighted Regularization

Here we consider another scheme for stabilizing the problem. Suppose F satisfies Assumption 6.5.1 and 6.5.3. For some $\lambda > 0$, define

$$\tilde{F}(x, x) = F(x, x) + \lambda R(x) \tag{6.14}$$

in which[8] $R(x)$ is an α-strongly convex regularization term such that $R(x) \geq 0, \forall x \in \mathcal{X}$ and $\min_{y \in \mathcal{X}} F(x, y) + \lambda R(y) = (1+\lambda) O(\tilde{\epsilon}_{\Pi,\pi^\star})$. For example, R can be $F(\pi^\star, \cdot)$ when $\pi^\star$ is (close) to optimal (e.g. in the case of DAGGER), or $R(x) = \mathbb{E}_{s \sim d^{\pi^\star}} \mathbb{E}_{a \sim \pi|s} \mathbb{E}_{a^* \sim \pi^\star|s}[d(a, a^*)]$, where π is a policy parametrized by x and $d(\cdot, \cdot)$ is some metric of space $\mathcal{A}$ (i.e. it uses the distance between π and $\pi^\star$ as regularization).

It can be seen that $\tilde{F}$ is uniformly $(1+\lambda)\alpha$-strongly convex in the second argument and $\nabla_2 \tilde{F}$ is uniformly β-continuous in the second argument. That is, if we choose $\lambda > \theta - 1$, then the stability constant $\tilde{\theta}$ of $\tilde{F}$ satisfies $\tilde{\theta} < 1$.

Corollary 6.6.2. *Define $\Delta_N = \frac{(\tilde{\theta} e^{1-\tilde{\theta}} G_2)^2}{2\alpha} N^{2(\tilde{\theta}-1)}$. Running* AGGREVATTE *with $\tilde{F}$ in* (6.14) *as the per-round loss has performance satisfies: for all $N > 0$,*

$$F(x_N, x_N) \leq (1 + \lambda) \left(O(\tilde{\epsilon}_{\Pi,\pi^\star}) + \Delta_N \right)$$

Proof. Because $F(x_N, x_N) = \tilde{F}(x_N, x_N) - \lambda R(x_N)$, the inequality can be proved by

[8] See Section 6.D for discussion of the case where $R(\cdot) = F(\pi^\star, \cdot)$ regardless of the condition $R(x) \geq 0$.

applying Theorem 6.5.2 to $\tilde{F}(x_N, x_N)$. ■

By Corollary 6.6.2, using AGGREVATTE to solve a weighted regularized problem in (6.14) would generate a convergent sequence for λ large enough. Unlike using a mixing policy, here the performance guarantee on the last policy is only worsened by a multiplicative constant on $\tilde{\epsilon}_{\Pi,\pi^\star}$, which can be made small by choosing a larger policy class.

The result in Corollary 6.6.2 can be strengthened particularly when

$$R(x) = \mathbb{E}_{s\sim d^{\pi^\star}} \mathbb{E}_{a\sim\pi|s} \mathbb{E}_{a^*\sim\pi^\star|s}[d(a, a^*)]$$

is used. In this case, it can be shown that $CR(x) \geq F(x, x)$ for some $C > 0$ (usually $C > 1$) (Pan et al., 2017a). That is, $F(x, x) + \lambda R(x) \geq (1 + \lambda/C)F(x, x)$. Thus, the multiplicative constant in Corollary 6.6.2 can be reduced from $1 + \lambda$ to $\frac{1+\lambda}{1+\lambda/C}$. It implies that simply by adding a portion of demonstrations gathered under the expert's distribution so that the leaner can anchor itself to the expert while minimizing $F(x, x)$, one does not have to find the best policy in the sequence $\{\pi_n\}_{n=1}^N$ as in (6.8), but just return the last policy π_N.

6.7 Conclusion

We contribute a new analysis of value aggregation, unveiling several interesting theoretical insights. Under a weaker assumption than the classical result, we prove that the convergence of the last policy depends solely on a problem's structural property and we provide a tight non-asymptotic bound on its performance in both deterministic and stochastic problems. In addition, using the new theoretical results, we show that the stability of the last policy can be reinforced by additional regularization with minor performance loss. This suggests that under proper conditions a practitioner can just run AGGREVATTE and then take the last policy, without performing an additional statistical test to find the best policy, as required by the classical analysis. Finally, as our results concerning the last policy are

based on the perturbation of gradients, we believe this provides a potential explanation as to why AGGREVATTE has demonstrated empirical success in non-convex problems with neural-network policies.

6.A Missing Proofs

6.A.1 Proof of Proposition 6.3.1

Proposition 6.3.1. *Suppose Π consists of deterministic linear policies (i.e. $a = \phi(s)^\top x$ for some feature map $\phi(s)$ and weight x) and $\forall s \in \mathcal{S}$, $c(s, \cdot)$ is strongly convex. Assumption 6.3.2 holds under any of the following when the state distribution is diverse enough:*

1. *$V^{\pi^\star}(s)$ is constant over $\mathcal{S}$ (in this case $A^{\pi^\star}(s, a)$ is equivalent to $c(s, a)$ up to a constant in a)*

2. *The problem is continuous-time and the dynamics are affine in action.*

Proof. Let π be parametrized by x. We prove the sufficient conditions by showing that $A^{\pi^\star}(s, a)$ is strongly convex in a for all $s \in \mathcal{S}$, which by the linear policy assumption implies $l_n(\pi)$ is strongly convex in x, provided that the state distribution is diverse enough.

For the first case, since $Q^{\pi^\star}(s, a) = c(s, a) + \mathbb{E}_{s'|s,a}[V^{\pi^\star}(s')]$, given the constant assumption, it follows that

$$A^{\pi^\star}(s, a) = Q^{\pi^\star}(s, a) - V^{\pi^\star}(s) = c(s, a) + \text{const.}$$

is strongly convex in terms of a.

For the second case, consider a system $ds = (f(s) + g(s)a)\, dt + h(s)dw$, where f, g, h are some matrix functions and dw is a Wiener process. By Hamilton-Jacobi-Bellman equation (Bertsekas et al., 1995), the advantage function can be written as

$$A^{\pi^\star}(s, a) = c(s, a) + \partial_s V^{\pi^\star}(s)^\top g(s)a + r(s)$$

where $r(s)$ is some function in s. Therefore, $A^{\pi^\star}(s,a)$ is strongly convex in a. ■

6.A.2 Proof of Theorem 6.5.1

Theorem 6.5.1. *Under Assumption 6.5.1 and 6.5.2,* AGGREVATTE *generates a sequence such that, for all* $N \geq 1$*,*

$$F(\hat{x}_N, \hat{x}_N) \leq \frac{1}{N}\sum_{n=1}^{N} l_n(x_n) \leq \epsilon_{\Pi,\pi^\star} + \frac{G_2^2}{2\alpha}\frac{\ln(N)+1}{N}$$

where $\hat{x}_N \coloneqq \arg\min_{x\in\{x_n\}_{n=1}^N} F(x,x)$.

Proof. The proof is based on a basic perturbation lemma in convex analysis (Lemma 6.A.1), which for example can be found in (McMahan, 2017), and a lemma for online learning (Lemma 6.A.2).

Lemma 6.A.1. *Let* $\phi_1 : \mathbb{R}^d \mapsto \mathbb{R}\bigcup\{\infty\}$ *be a convex function such that* $x_1 = \arg\min_x \phi_t(x)$ *exits. Let* ψ *be a function such that* $\phi_2(x) = \phi_1(x) + \psi(x)$ *is* α*-strongly convex with respect to* $\|\cdot\|$*. Let* $x_2 = \arg\min_x \phi_2(x)$*. Then, for any* $g \in \partial\psi(x_1)$*, we have*

$$\|x_1 - x_2\| \leq \frac{1}{\alpha}\|g\|_*$$

and for any x'

$$\phi_2(x_1) - \phi_2(x') \leq \frac{1}{2\alpha}\|g\|_*^2$$

When ϕ_1 *and* ψ *are quadratics (with* ψ *possibly linear) the above holds with equality.*

Lemma 6.A.2. *Let* $l_n(x)$ *be a sequence of functions. Denote* $l_{1:n}(x) = \sum_{\tau=1}^{n} l_\tau(x)$*. and let*

$$x_n^* = \arg\min_{x\in K} l_{1:n}(x)$$

Then for any sequence $\{x_1, \ldots, x_N\}$, $\tau \geq 1$, *and any* $x^* \in K$, *it holds*

$$\sum_{n=\tau}^{N} l_n(x_n) \leq l_{1:N}(x_N^*) - l_{1:\tau-1}(x_{\tau-1}^*) + \sum_{n=\tau}^{N} l_{1:n}(x_n) - l_{1:n}(x_n^*)$$

Proof. Introduce a slack loss function $l_0(\cdot) = 0$ and define $x_0^* = 0$ for index convenience. This does not change the optimum, since $l_{0:n}(x) = l_{1:n}(x)$.

$$\begin{aligned}
\sum_{n=\tau}^{N} l_n(x_n) &= \sum_{n=\tau}^{N} l_{0:n}(x_n) - l_{0:n-1}(x_n) \\
&\leq \sum_{n=\tau}^{N} l_{0:n}(x_n) - l_{0:n-1}(x_{n-1}^*) \\
&= l_{0:N}(x_N^*) - l_{0:\tau-1}(x_{\tau-1}^*) + \sum_{n=\tau}^{N} l_{0:n}(x_n) - l_{0:n}(x_n^*)
\end{aligned}$$

■

Note Lemma 6.A.2 does not require l_n to be convex and the minimum to be unique.

To prove Theorem 6.5.1, we first note that by definition of $\hat{x}_N$, it satisfies $F(\hat{x}_N, \hat{x}_N) \leq \frac{1}{N}\sum_{n=1}^{N} l_n(x_n)$. To bound the average performance, we use Lemma 6.A.2 and write

$$\sum_{n=1}^{N} l_n(x_n) \leq l_{1:N}(x_{N+1}) + \sum_{n=1}^{N} l_{1:n}(x_n) - l_{1:n}(x_{n+1})$$

since $x_n = \arg\min_{x \in \mathcal{X}} l_{1:n-1}(x)$. Then because $l_{1:k}$ is $k\alpha$-strongly convex, by Lemma 6.A.1,

$$\sum_{n=1}^{N} l_n(x_n) \leq l_{1:N}(x_n^*) + \sum_{n=1}^{N} \frac{\|\nabla l_n(x_n)\|_*^2}{2\alpha n}.$$

Finally, dividing the upper-bound by n and using the facts that $\sum_{k=1}^{n} \frac{1}{k} \leq \ln(n) + 1$ and $\min a_i \leq \frac{1}{n}\sum a_i$ for any scalar sequence $\{a_n\}$, we have the desired result. ■

6.A.3 Proof of Theorem 6.5.3

Theorem 6.5.3. *There is a problem such that running* AGGREVATTE *for* N *iterations results in* $F(x_N, x_N) \geq \tilde{\epsilon}_{\Pi,\pi^\star} + \Omega(N^{2(\theta-1)})$. *In particular, if* $\theta > 1$, *the policy sequence*

and performance sequence diverge.

Proof. Consider the example in Section 6.4. For this problem, $T = 2$, $J(x^*) = 0$, and $\tilde{\epsilon}_{\Pi,\pi^\star} = 0$, implying $F(x,x) = \frac{1}{2}J(x) = \frac{1}{2}(\theta-1)^2x^2$. Therefore, to prove the theorem, we focus on the lower bound of x_N^2.

Since $x_n = \arg\min_{x\in\mathcal{X}} l_{1:n-1}(x)$ and the cost is quadratic, we can write

$$\begin{aligned} x_{n+1} &= \arg\min_{x\in\mathcal{X}} l_{1:n}(x) \\ &= \arg\min_{x\in\mathcal{X}} (n-1)(x-x_n)^2 + (x-\theta x_n)^2 \\ &= (1-\frac{1-\theta}{n})x_n \end{aligned}$$

If $\theta = 1$, then $x_N = x_1$ and the bound holds trivially. For general cases, let $p_n = \ln(x_n^2)$.

$$p_N - p_2 = 2\sum_{n=2}^{N-1} \ln\left(1-\frac{1-\theta}{n}\right) \geq -2(1-\theta)\sum_{n=2}^{N-1}\frac{1}{n-(1-\theta)}$$

where the inequality is due to the fact that $\ln(1-x) \geq \frac{-x}{1-x}$ for $x < 1$. We consider two scenarios. Suppose $\theta < 1$.

$$\begin{aligned} p_N - p_2 &\geq -2(1-\theta)\int_1^{N-1} \frac{1}{x-(1-\theta)}dx \\ &= -2(1-\theta)\ln(x-(1-\theta))|_1^{N-1} \\ &= -2(1-\theta)\left(\ln(N+\theta-2) - \ln(\theta)\right) \\ &\geq -2(1-\theta)\ln(N+\theta-2) \end{aligned}$$

Therefore, $x_N^2 \geq x_2^2(N+\theta-2)^{2(\theta-1)} \geq \Omega(N^{2(\theta-1)})$.

On the other hand, suppose $\theta > 1$.

$$\begin{aligned} p_N - p_2 &\geq 2(\theta-1)\int_2^N \frac{1}{x-(1-\theta)}dx \\ &= 2(\theta-1)\ln(x-(1-\theta))|_2^N \\ &= 2(\theta-1)\left(\ln(N-1+\theta) - \ln(1+\theta)\right) \end{aligned}$$

Therefore, $x_N^2 \geq x_2^2(N-1+\theta)^{2(\theta-1)}(1+\theta)^{-2(\theta-1)} \geq \Omega(N^{2(\theta-1)})$. Substituting the lower bound on x_N^2 into the definition of $F(x,x)$ concludes the proof. ■

6.A.4 Proof of Corollary 6.5.1

Corollary 6.5.1. *Under the assumptions in Lemma 6.5.1, for* $n > m$, $S_{m:n} \leq O(\frac{\theta}{(n-m)m^{2-\theta}} + \frac{1}{n^{1-\theta}})$.

Proof. To prove the corollary, we introduce a basic lemma

Lemma 6.A.3. *(Lan, 2013, Lemma 1) Let* $\gamma_k \in (0,1)$, $k = 1, 2, \ldots$ *be given. If the sequence* $\{\Delta_k\}_{k\geq 0}$ *satisfies*

$$\Lambda_{k+1} \leq (1-\gamma_k)\Lambda_k + B_k,$$

then

$$\Lambda_k \leq \Gamma_k + \Gamma_k \sum_{i=1}^{k} \frac{B_i}{\Gamma_{i+1}}$$

where $\Gamma_1 = \Lambda_1$ *and* $\Gamma_{k+1} = (1-\gamma_k)\Gamma_k$.

To bound the sequence $S_{m:n+1}$, we first apply Lemma 6.5.1. Fixed m, for any $n \geq$

$m+1$, we have

$$\begin{aligned} S_{m:n+1} &\leq \left(1 - \frac{1}{n-m+1}\right) S_{m:n} + \|x_{n+1} - x_n\| \\ &\leq \left(1 - \frac{1}{n-m+1}\right) S_{m:n} + \frac{\theta}{n} S_n \\ &\leq \left(1 - \frac{1}{n-m+1}\right) S_{m:n} + \frac{\theta c}{n^{2-\theta}} \end{aligned}$$

where $c = S_2 e^{1-\theta}$.

Then we apply Lemma 6.A.3. Let $k = n - m + 1$ and define $R_k = S_{m:m+k-1} = S_{m:n}$ for $k \geq 2$. Then we rewrite the above inequality as

$$R_{k+1} \leq \left(1 - \frac{1}{k}\right) R_k + \frac{\theta c}{(k+m-1)^{2-\theta}}$$

and define

$$\Gamma_k \coloneqq \begin{cases} 1, & k = 1 \\ (1 - \frac{1}{k-1})\Gamma_{k-1}, & k \geq 2 \end{cases}$$

By Proposition 6.5.1, the above conversion implies for some positive constant c,

$$R_2 = S_{m:m+1} = \|x_{m+1} - x_m\| \leq \frac{\theta S_m}{m} \leq \frac{\theta c}{m^{2-\theta}}$$

and $\Gamma_k \leq O(1/k)$ and $\frac{\Gamma_k}{\Gamma_i} \leq O(\frac{i}{k})$. Thus, by Lemma 6.A.3, we can derive

$$\begin{aligned} R_k &\leq \frac{1}{k} R_2 + O\left(\theta c \sum_{i=1}^{k} \frac{i}{k} \frac{1}{(i+m-1)^{2-\theta}}\right) \\ &\leq \frac{1}{k} R_2 + O\left(\frac{\theta c}{k} \frac{k}{\theta} \frac{1}{(m+k-1)^{1-\theta}}\right) \\ &= \frac{1}{k} R_2 + O\left(\frac{1}{(m+k-1)^{1-\theta}}\right) \\ &\leq \frac{1}{k} \frac{\theta c}{m^{2-\theta}} + O\left(\frac{1}{(m+k-1)^{1-\theta}}\right) = O(\frac{1}{n^{1-\theta}}) \end{aligned}$$

where we use the following upper bound in the second inequality

$$\begin{aligned}
\sum_{i=1}^{k} \frac{i}{(i+m-1)^{2-\theta}} &\leq \int_0^k \frac{x}{(x+m-1)^{2-\theta}} dx \\
&= \frac{m+(1-\theta)x-1}{\theta(1-\theta)(m+x-1)^{1-\theta}}\Big|_0^k \\
&= \frac{(1-\theta)k+m-1}{\theta(1-\theta)(m+k-1)^{1-\theta}} - \frac{m-1}{\theta(1-\theta)(m-1)^{1-\theta}} \\
&= \frac{k}{\theta}\frac{1}{(m+k-1)^{1-\theta}} + \frac{m-1}{\theta(1-\theta)}\left(\frac{1}{(m+k-1)^{1-\theta}} - \frac{1}{(m-1)^{1-\theta}}\right) \\
&\leq \frac{k}{\theta}\frac{1}{(m+k-1)^{1-\theta}}
\end{aligned}$$

■

6.A.5 Proof of Lemma 6.6.1

Lemma 6.6.1. *Let $\|p_1 - p_2\|_1$ denote the total variational distance between distributions p_1 and p_2. Assume*[9] *for any policy π, π' parameterized by x, y it satisfies $\frac{1}{T}\sum_{t=0}^{T-1} \|d_t^{\pi} - d_t^{\pi'}\|_1 \leq \frac{\beta}{2G_2}\|x-y\|$ and assume $\|\nabla_x \mathbb{E}_{\pi}[A^{\pi^\star}](s)\|_* \leq G_2$. Then $\nabla_2 F$ is uniformly $(1-q^T)\beta$-Lipschitz continuous in the second argument.*

Proof. Define δ_t^{π} such that $d_{;q}^{\pi}(s) = (1-q^t)\delta_t^{\pi}(s) + q^t d^{\pi^\star}(s)$, and define $g_t^z(s) = \nabla_z \mathbb{E}_{\pi}[Q^{\pi^\star}](s)$, for π parametrized by z; then by assumption, $\|g_t^z\|_* < G_2$. Let π, π' be two policies pa-

[9] These two are sufficient to Assumption 6.5.1.2 and 6.5.3.

rameterized by $x, y \in \mathcal{X}$, respectively. Then

$$\begin{aligned}
\|\nabla_2 \hat{F}(x,z) - \nabla_2 \hat{F}(y,z)\|_* &= \|\mathbb{E}_{d^{\tilde{\pi}}}[g_t^z] - \mathbb{E}_{d^{\tilde{\pi}'}}[g_t^z]\|_* \\
&= \|\frac{1}{T}\sum_{t=0}^{T-1}(1-q^t)(\mathbb{E}_{\delta_{\pi|t;q}}[g_t^z] - \mathbb{E}_{\delta_{\pi'|t;q}}[g_t^z])\|_* \\
&\leq (1-q^T)\frac{1}{T}\sum_{t=0}^{T-1}\|\mathbb{E}_{\delta_{\pi|t;q}}[g_t^z] - \mathbb{E}_{\delta_{\pi'|t;q}}[g_t^z]\|_* \\
&\leq (1-q^T)\frac{2G_2}{T}\sum_{t=0}^{T-1}\|\delta_{\pi|t;q} - \delta_{\pi'|t;q}\|_1 \\
&\leq (1-q^T)\frac{2G_2}{T}\sum_{t=0}^{T-1}\|d_t^{\pi} - d_t^{\pi'}\|_1 \\
&\leq (1-q^T)\beta\|x-y\|
\end{aligned}$$

in which the second to the last inequality is because the divergence between d_t^{π} and $d_t^{\pi'}$ is the largest among all state distributions generated by the mixing policies. ■

6.B Analysis of AGGREVATTE in Stochastic Problems

Here we give the complete analysis of the convergence of AGGREVATTE in stochastic problems using finite-sample approximation. For completeness, we restate the results below: Let $f(x;s) = \mathbb{E}_{\pi}[A^{\pi^\star}]$ (i.e. $l_n(x) = \mathbb{E}_{d^{\pi_n}}[f(x;s)]$, where policy π is a policy parametrized by x. Instead of using $l_n(\cdot)$ as the per-round loss in the nth iteration, we use consider its finite samples approximation $g_n(\cdot) = \sum_{k=1}^{m_n} f(\cdot; s_{n,k})$, where m_n is the number of independent samples collected in the nth iteration.

Theorem 6.5.4. *In addition to Assumptions 6.5.3 and 6.5.4, assume $f(x;s)$ is α-strongly convex in x and $\|\nabla f(x;s)\|_* < G_2$ almost surely. Let $\theta = \frac{\beta}{\alpha}$ and suppose $m_n = m_0 n^r$ for some $r \geq 0$. For all $N > 0$, with probability at least $1-\delta$,*

$$F(x_N, x_N) \leq \tilde{\epsilon}_{\Pi,\pi^\star} + \tilde{O}\left(\frac{\theta^2}{c}\frac{\ln(1/\delta) + C_{\mathcal{X}}}{N^{\min\{r,2,2-2\theta\}}}\right) + \tilde{O}\left(\frac{\ln(1/\delta) + C_{\mathcal{X}}}{cN^{\min\{2,1+r\}}}\right)$$

where $c = \frac{\alpha}{G_2^2 m_0}$ *and* $C_{\mathcal{X}}$ *is a constant*[10] *of the complexity of* Π.

6.B.1 Uniform Convergence of Vector-Valued Martingales

To prove Theorem 6.5.4, we first introduces several concentration inequalities of vector-valued martingales by (Hayes, 2005) in Section 6.B.1. Then we prove some basic lemmas regarding the convergence the stochastic dynamical systems of $\nabla g_n(x)$ specified by AGGREVATTE in Section 6.B.1 and 6.B.1. Finally, the lemmas in these two sections are extended to provide uniform bounds, which are required to prove Theorem 6.5.4. In this section, we will state the results generally without limiting ourselves to the specific functions used in AGGREVATTE.

Generalization of Azuma-Hoeffding Lemma

First we introduce two theorems by Hayes (2005) which extend Azuma-Hoeffding lemma to vector-valued martingales but without dependency on dimension.

Theorem 6.B.1. *(Hayes, 2005, Theorem 1.8) Let* $\{X_n\}$ *be a (very-weak) vector-valued martingale such that* $X_0 = 0$ *and for every* n, $\|X_n - X_{n-1}\| \leq 1$ *almost surely. Then, for every* $a > 0$*, it holds*

$$\Pr(\|X_n\| \geq a) < 2e \exp\left(\frac{-(a-1)^2}{2n}\right)$$

Theorem 6.B.2. *(Hayes, 2005, Theorem 7.4) Let* $\{X_n\}$ *be a (very-weak) vector-valued martingale such that* $X_0 = 0$ *and for every* n, $\|X_n - X_{n-1}\| \leq c_n$ *almost surely. Then, for every* $a > 0$*, it holds*

$$\Pr(\|X_n\| \geq a) < 2 \exp\left(\frac{-(a-Y_0)^2}{2\sum_{i=1}^n c_i^2}\right)$$

[10]The constant $C_{\mathcal{X}}$ can be thought as $\ln|\mathcal{X}|$, where $|\mathcal{X}|$ measures the size of $\mathcal{X}$ in e.g. Rademacher complexity or covering number (Mohri, Rostamizadeh, and Talwalkar, 2012). For example, $\ln|\mathcal{X}|$ can be linear in $\dim \mathcal{X}$.

where $Y_0 = \max\{1 + \max c_i, 2\max c_i\}$.

Concentration of i.i.d. Vector-Valued Functions

Theorem 6.B.1 immediately implies the concentration of approximating vector-valued functions with finite samples.

Lemma 6.B.1. *Let* $x \in \mathcal{X}$ *and let* $f(x) = \mathbb{E}_\omega[f(x;\omega)]$, *where* $f : \mathcal{X} \to E$ *and* E *is equipped with norm* $\|\cdot\|$. *Assume* $\|f(x;\omega)\| \leq G$ *almost surely. Let* $g(x) = \frac{1}{M}\sum_{m=1}^{M} f(x;\omega_k)$ *be its finite sample approximation. Then, for all* $\epsilon > 0$,

$$\Pr(\|g(x) - f(x)\| \geq \epsilon) < 2e \exp\left(-\frac{(\frac{M\epsilon}{2G} - 1)^2}{2M}\right)$$

In particular, for $0 < \epsilon \leq 2G$,

$$\Pr(\|g(x) - f(x)\| \geq \epsilon) < 2e^2 \exp\left(-\frac{M\epsilon^2}{8G^2}\right)$$

Proof. Define $X_m = \frac{1}{2G}\sum_{k=1}^{m} f(x;\omega_k) - f(x)$. Then X_m is vector-value martingale and $\|X_m - X_{m-1}\| \leq 1$. By Theorem 6.B.1,

$$\Pr(\|g(x) - f(x)\| \geq \epsilon) = \Pr(\|X_M\| \geq \frac{M\epsilon}{2G}) < 2e \exp\left(-\frac{(\frac{M\epsilon}{2G} - 1)^2}{2M}\right)$$

Suppose $\frac{\epsilon}{2G} < 1$. Then $\Pr(\|X_M\| \geq \epsilon) < 2e^2 \exp\left(-\frac{M\epsilon^2}{8G^2}\right)$. ■

Concentration of the Stochastic Process of AGGREVATTE

Here we consider a stochastic process that shares the same characteristics of the dynamics of $\frac{1}{n}\nabla g_{1:n}(x)$ in AGGREVATTE and provide a lemma about its concentration.

Lemma 6.B.2. *Let* $n = 1 \ldots N$ *and* $\{m_i\}$ *be a non-decreasing sequence of positive integers. Given* $x \in \mathcal{X}$, *let* $Y_n := \{v_n(x;\omega_{n,k})\}_{k=1}^{m_n}$ *be a set of random vectors in some*

normed space with norm $\|\cdot\|$ *defined as follows: Let* $Y_{1:n} \coloneqq \{Y_k\}_{k=1}^n$. *Given* $Y_{1:n-1}$, $\{v_n(x;\omega_{n,k})\}_{k=1}^{m_n}$ *are* m_n *independent random vectors such that* $v_n(x) \coloneqq \mathbb{E}_\omega[v_n(x;\omega)|Y_{1:n-1}]$ *and* $\|v_n(x;\omega)\| \leq G$ *almost surely. Define* $g_n(x) \coloneqq \frac{1}{m_n}\sum_{k=1}^{m_n} v_n(x;\omega_{n,k})$, *and let* $\bar{g}_n = \frac{1}{n}g_{1:n}$ *and* $\bar{v}_n = \frac{1}{n}v_{1:n}$. *Then for all* $\epsilon > 0$,

$$\Pr(\|\bar{g}_n(x) - \bar{v}_n(x)\| \geq \epsilon) < 2\exp\left(\frac{-(nM^*\epsilon - Y_0)^2}{8G^2M^{*2}\sum_{i=1}^n \frac{1}{m_i}}\right)$$

in which $M^* = \prod_{i=1}^n m_i$ *and* $Y_0 = \max\{1 + \frac{2M^*G}{m_0}, 2\frac{2M^*G}{m_0}\}$. *In particular, if* $\frac{2M^*G}{m_0} > 1$, *for* $0 < \epsilon \leq \frac{Gm_0}{n}\sum_{i=1}^n \frac{1}{m_i}$,

$$\Pr(\|\bar{g}_n(x) - \bar{v}_n(x)\| \geq \epsilon) < 2e\exp\left(\frac{-n^2\epsilon^2}{8G^2\sum_{i=1}^n \frac{1}{m_i}}\right)$$

Proof. Let $M = \sum_{i=1}^n m_i$. Consider a martingale, for $m = l + \sum_{i=1}^{k-1} m_i$,

$$X_m = \frac{M^*}{m_k}\sum_{i=1}^{l} v_k(x;\omega_{k,i}) - v_k(x) + \sum_{i=1}^{k-1}\frac{M^*}{m_i}\sum_{j=1}^{m_i} v_i(x;\omega_{i,j}) - v_i(x).$$

That is, $X_M = nM^*(\bar{g}_n - \bar{v}_n)$ and $\|X_m - X_{m-1}\| \leq \frac{2M^*G}{m_i}$ for some appropriate m_i. Applying Theorem 6.B.2, we have

$$\Pr(\|\bar{g}_n - \bar{v}_n\| \geq \epsilon) = \Pr(\|X_M\| \geq nM^*\epsilon) < 2\exp\left(\frac{-(nM^*\epsilon - Y_0)^2}{2\sum_{m=1}^M c_m^2}\right)$$

where

$$\sum_{m=1}^M c_m^2 = \sum_{i=1}^n\sum_{j=1}^{m_i}\left(\frac{2GM^*}{m_i}\right)^2 = 4G^2M^{*2}\sum_{i=1}^n\frac{1}{m_i}.$$

In addition, by assumption $m_i \leq m_{i-1}$, $Y_0 = \max\{1 + \frac{2M^*G}{m_0}, 2\frac{2M^*G}{m_0}\}$. This gives the first inequality.

For the special case, the following holds

$$\frac{-(nM^*\epsilon - Y_0)^2}{2\sum_{m=1}^{M} c_m^2} = \frac{-n^2M^{*2}\epsilon^2}{8G^2M^{*2}\sum_{i=1}^{n}\frac{1}{m_i}} + \frac{2nM^*\epsilon Y_0 - Y_0^2}{8G^2M^{*2}\sum_{i=1}^{n}\frac{1}{m_i}} \leq \frac{-n^2\epsilon^2}{4G^2\sum_{i=1}^{n}\frac{1}{m_i}} + 1$$

if ϵ satisfies

$$2nM^*\epsilon Y_0 < 8G^2M^{*2}\sum_{i=1}^{n}\frac{1}{m_i} \implies \epsilon < \frac{4G^2M^*}{Y_0 n}\sum_{i=1}^{n}\frac{1}{m_i}$$

Substituting the condition that $Y_0 = \frac{4M^*G}{m_0}$ when $\frac{2M^*G}{m_0} > 1$, a sufficient range of ϵ can be obtained as

$$\frac{4G^2M^*}{Y_0 n}\sum_{i=1}^{n}\frac{1}{m_i} = \frac{Gm_0}{n}\sum_{i=1}^{n}\frac{1}{m_i} \geq \epsilon.$$

■

Uniform Convergence

The above inequality holds for a particular $x \in \mathcal{X}$. Here we use the concept of covering number to derive uniform bounds that holds for all $x \in \mathcal{X}$. (Similar (and tighter) uniform bounds can also be derived using Rademacher complexity.)

Definition 6.B.1. Let S be a metric space and $\eta > 0$. The covering number $\mathcal{N}(S, \eta)$ is the minimal $l \in \mathcal{N}$ such that S is is covered by l balls of radius η. When S is compact, $\mathcal{N}(S, \eta)$ is finite.

As we are concerned with vector-valued functions, let E be a normed space with norm $\|\cdot\|$. Consider a mapping $f : \mathcal{X} \to \mathcal{B}$ defined as $f : x \mapsto f(x, \cdot)$, where $\mathcal{B} = \{g : \Omega \to E\}$ is a Banach space of vector-valued functions with norm $\|g\|_{\mathcal{B}} = \sup_{\omega\in\Omega}\|g(\omega)\|$. Assume $\mathcal{B}_{\mathcal{X}} = \{f(x, \cdot) : x \in \mathcal{X}\}$ is a compact subset in $\mathcal{B}$. Then the covering number of $\mathcal{H}$ is finite and given as $\mathcal{N}(\mathcal{B}_{\mathcal{X}}, \eta)$. That is, there exists a finite set $\mathcal{C}_{\mathcal{X}} = \{x_i \in \mathcal{X}\}_{i=1}^{\mathcal{N}(\mathcal{B}_{\mathcal{X}},\eta)}$ such that $\forall x \in \mathcal{X}, \min_{y\in\mathcal{C}_{\mathcal{X}}}\|f(x, \cdot) - f(y, \cdot)\|_{\mathcal{B}} < \eta$.

Usually, the covering is a polynomial function of η. For example, suppose $\mathcal{X}$ is a ball of radius R in a d-dimensional Euclidean space, and f is L-Lipschitz in x (i.e. $\|f(x,\cdot) - f(y,\cdot)\|_{\mathcal{B}} \leq L\|x-y\|$). Then (Cucker and Zhou, 2007) $\mathcal{N}(\mathcal{B}_{\mathcal{X}}, \eta) \leq \mathcal{N}(\mathcal{X}, \frac{\eta}{L}) \leq \left(\frac{2RL}{\eta} + 1\right)^d$. Therefore, henceforth we will assume

$$\ln \mathcal{N}(\mathcal{B}_X X, \eta) \leq C_{\mathcal{X}} \ln(\frac{1}{\eta}) < \infty \tag{6.15}$$

for some constant $C_{\mathcal{X}}$ independent of η, which characterizes the complexity of $\mathcal{X}$.

Using covering number, we derive uniform bounds for the lemmas in Section 6.B.1 and 6.B.1.

Lemma 6.B.3. *Under the assumptions in Lemma 6.B.1, for* $0 < \epsilon \leq 2G$*,*

$$\Pr(\sup_{x\in\mathcal{X}} \|g(x) - f(x)\| \geq \epsilon) < 2e^2 \mathcal{N}(\mathcal{B}_{\mathcal{X}}, \frac{\epsilon}{4}) \exp\left(-\frac{M\epsilon^2}{32G^2}\right)$$

Proof. Choose $\mathcal{C}_{\mathcal{X}}$ be the set of the centers of the covering balls such that $\forall x \in \mathcal{X}$, $\min_{y\in\mathcal{C}_{\mathcal{X}}} \|f(x,\cdot) - f(y,\cdot)\|_{\mathcal{B}} < \eta$. Since $f(x) = \mathbb{E}_\omega[f(x,\omega)]$, it also holds $\min_{y\in\mathcal{C}_{\mathcal{X}}} \|f(x) - f(y)\| < \eta$. Let B_y be the η-ball centered for $y \in \mathcal{C}_{\mathcal{X}}$. Then

$$\begin{aligned}
\sup_{y\in\mathcal{X}} \|g(x) - f(x)\| &\leq \max_{y\in\mathcal{C}_{\mathcal{X}}} \sup_{x\in B_y} \|g(x) - g(y)\| + \|g(y) - f(y)\| + \|f(y) - f(x)\| \\
&\leq \max_{y\in\mathcal{C}_{\mathcal{X}}} \|g(y) - f(y)\| + 2\eta
\end{aligned}$$

Choose $\eta = \frac{\epsilon}{4}$ and then it follows that

$$\sup_{x\in\mathcal{X}} \|g(x) - f(x)\| \geq \epsilon \implies \max_{y\in\mathcal{C}_{\mathcal{X}}} \|g(y) - f(y)\| \geq \frac{\epsilon}{2}$$

The final result can be obtained by first for each $y \in \mathcal{C}_{\mathcal{X}}$ applying the concentration inequality with $\epsilon/2$ and then a uniform bound over $\mathcal{C}_{\mathcal{X}}$. ■

Similarly, we can give a uniform version of Lemma 6.B.2.

Lemma 6.B.4. *Under the assumptions in Lemma 6.B.2, if* $\frac{2M^*G}{m_0} > 1$, *for* $0 < \epsilon \leq \frac{Gm_0}{n}\sum_{i=1}^{n}\frac{1}{m_i}$ *and for a fixed* $n \geq 0$,

$$\Pr(\sup_{x\in\mathcal{X}} \|\bar{g}_n(x) - \bar{l}_n(x)\| \geq \epsilon) < 2e\mathcal{N}\left(\mathcal{B}_{\mathcal{X}}, \frac{\epsilon}{4}\right)\exp\left(\frac{-n^2\epsilon^2}{32G^2\sum_{i=1}^{n}\frac{1}{m_i}}\right)$$

6.B.2 Proof of Theorem 6.5.4

We now refine Lemma 6.5.1 and Proposition 6.5.1 to prove the convergence of AGGREVATTE in stochastic problems. We use $\bar{\cdot}$ to denote the average (e.g. $\bar{l}_n = \frac{1}{n}l_{1:n}$.)

Bound on $\|x_{n+1} - x_n\|$

First, we show the error due to finite-sample approximation.

Lemma 6.B.5. *Let* $\xi_n = \nabla l_n - \nabla g_n$. *Running* AGGREVATTE *with* $g_n(\cdot)$ *as per-round loss gives, for* $n \geq 2$,

$$\|x_{n+1} - x_n\| \leq \frac{\theta S_n}{n} + \frac{1}{n\alpha}\left(\|\xi_n(x_n)\|_* + \|\bar{\xi}_{n-1}(x_n)\|_*\right)$$

Proof. Because $g_{1:n}(x)$ is $n\alpha$-strongly convex in x, we have

$$\begin{aligned}
n\alpha\|x_{n+1} - x_n\|^2 &\leq \langle \nabla g_n(x_n), x_n - x_{n+1}\rangle \\
&\leq \langle \nabla g_n(x_n) - \nabla\bar{g}_{n-1}(x_n), x_n - x_{n+1}\rangle \\
&\leq \|\nabla l_n(x_n) - \nabla\bar{l}_{n-1}(x_n)\|_*\|x_n - x_{n+1}\| \\
&\quad + \|\nabla l_n(x_n) - \nabla g_n(x_n) - \nabla\bar{l}_{n-1}(x_n) + \nabla\bar{g}_{n-1}(x_n)\|_*\|x_n - x_{n+1}\|
\end{aligned}$$

where the second inequality is because $x_n = \arg\min_{x\in\mathcal{X}} g_{1:n-1}(x)$. Now we use the fact

that the smoothness applies to f (not necessarily to g) and derive the statement

$$\begin{aligned}\|x_{n+1} - x_n\| &\leq \frac{\theta S_n}{n} + \frac{1}{n\alpha}\|\nabla l_n(x_n) - \nabla g_n(x_n) - \nabla \bar{l}_{n-1}(x_n) + \nabla \bar{g}_{n-1}(x_n)\|_* \\ &\leq \frac{\theta S_n}{n} + \frac{1}{n\alpha}\left(\|\xi_n(x_n)\|_* + \|\bar{\xi}_{n-1}(x_n)\|_*\right) \qquad \blacksquare\end{aligned}$$

Given the intermediate step in Lemma 6.B.5, we apply Lemma 6.B.1 to bound the norm of ξ_k and give the refinement of Lemma 6.5.1 for stochastic problems.

Lemma 6.B.6. *Suppose $m_n = m_0 n^r$ for some $r \geq 0$. Under previous assumptions, running* AGGREVATTE *with $g_n(\cdot)$ as per-round loss, the following holds with probability at least $1-\delta$: For a fixed $n \geq 2$,*

$$\|x_{n+1} - x_n\| \leq \frac{\theta S_n}{n} + O\left(\frac{G_2}{n\alpha\sqrt{m_0}}\left(\sqrt{\frac{\ln(1/\delta)}{n^{\min\{r,2\}}}} + \sqrt{\frac{C_{\mathcal{X}}/n}{n^{\min\{r,1\}}}}\right)\right)$$

where $C_{\mathcal{X}}$ is a constant depending on the complexity of $\mathcal{X}$ and the constant term in big-O is some universal constant.

Proof. To show the statement, we bound $\|\xi_n(x_n)\|_*$ and $\|\bar{\xi}_{1:n-1}(x_n)\|_*$ in Lemma 6.B.5 using the concentration lemmas derived in Section 6.B.1.

The First Term: To bound $\|\xi_n(x_n)\|_*$, because the sampling of ξ_n is independent of x_n, bounding $\|\xi_n(x_n)\|_*$ does not require a uniform bound. Here we use Lemma 6.B.1 and consider ϵ_1 such that

$$2e^2 \exp\left(-\frac{m_n \epsilon_1^2}{8G_2^2}\right) = \frac{\delta}{2} \implies \epsilon_1 = \sqrt{\frac{8G_2^2}{m_n}\ln\left(\frac{4e^2}{\delta}\right)} = O\left(\sqrt{\frac{G_2^2}{m_n}\ln\left(\frac{1}{\delta}\right)}\right) \tag{6.16}$$

Note we we used the particular range of ϵ in Lemma 6.B.1 for convenience, which is valid if we choose $m_0 > 2G_2 \ln\left(\frac{4e^2}{\delta}\right)$. This condition is not necessary; it is only used to simplify the derivation, and using a different range of ϵ would simply lead to a different constant.

The Second Term: To bound $\|\bar{\xi}_{n-1}(x_n)\|_*$, we apply a uniform bound using Lemma 6.B.4. For simplicity, we use the particular range $0 < \epsilon \leq \frac{G_2 m_0}{n} \sum_{i=1}^{n} \frac{1}{m_i}$ and assume $\frac{2M^* G_2}{m_0} > 1$ (which implies $Y_0 = \frac{4M^* G_2}{m_0}$) (again this is not necessary). We choose ϵ_2 such that

$$2e\mathcal{N}(\mathcal{B}_{\mathcal{X}}, \frac{\epsilon_2}{4}) \exp\left(\frac{-(n-1)^2\epsilon_2^2}{32G_2^2 \sum_{i=1}^{n-1} \frac{1}{m_i}}\right) \leq \frac{\delta}{2}$$
$$\implies \ln(2e) + \ln \mathcal{N}(\mathcal{B}_{\mathcal{X}}, \frac{\epsilon_2}{4}) + \frac{-(n-1)^2\epsilon_2^2}{32G_2^2 \sum_{i=1}^{n-1} \frac{1}{m_i}} \leq -\ln(\frac{2}{\delta})$$

Since $\ln \mathcal{N}(\mathcal{B}_{\mathcal{X}}, \frac{\epsilon_2}{4}) = C_{\mathcal{X}} \ln\left(\frac{4}{\epsilon_2}\right) \leq c_s C_{\mathcal{X}} \epsilon_2^{-s}$ for arbitrary $s > 0$ and some c_s, a sufficient condition can be obtained by solving for ϵ_2 such that

$$\frac{c_0}{\epsilon_2^s} - c_2\epsilon_2^2 = -c_1 \implies c_2\epsilon_2^{2+s} - c_1\epsilon_2^s - c_0 = 0$$

where $c_0 = c_s C_{\mathcal{X}}$, $c_2 = \frac{(n-1)^2}{32G_2^2 \sum_{i=1}^{n-1} \frac{1}{m_i}}$, and $c_1 = \ln(\frac{4e}{\delta})$. To this end, we use a basic lemma of polynomials.

Lemma 6.B.7. *Cucker and Zhou, 2007, Lemma 7.2 Let $c_1, c_2, \ldots, c_l > 0$ and $s > q_1 > q_2 > \cdots > q_{l-1} > 0$. Then the equation*

$$x^s - c_1 x^{q_1} - c_2 x^{q_2} - \cdots - c_{l-1} x^{q_{l-1}} - c_l = 0$$

has a unique solution x^*. In addition,

$$x^* \leq \max\left\{(lc_1)^{1/(s-q_1)}, (lc_2)^{1/(s-q_2)}, \ldots, (lc_{l-1})^{1/(s-q_{l-1})}, (lc_1)^{1/s}\right\}$$

Therefore, we can choose an ϵ_2 which satisfies

$$\begin{aligned}
\epsilon_2 &\leq \max\left\{\left(\frac{2c_1}{c_2}\right)^{1/2}, \left(\frac{2c_0}{c_2}\right)^{1/(2+s)}\right\} \\
&= \max\left\{\left(\frac{64\ln(\frac{4e}{\delta})G_2^2\sum_{i=1}^{n-1}\frac{1}{m_i}}{(n-1)^2}\right)^{1/2}, \left(\frac{64c_sC_{\mathcal{X}}G_2^2\sum_{i=1}^{n-1}\frac{1}{m_i}}{(n-1)^2}\right)^{1/(2+s)}\right\} \\
&\leq O\left(\sqrt{\left(C_{\mathcal{X}} + \ln\left(\frac{1}{\delta}\right)\right)\frac{G_2^2}{n^2}\sum_{i=1}^{n}\frac{1}{m_i}}\right)
\end{aligned}$$

Error Bound Suppose $m_n = m_0 n^r$, for $r \geq 0$. Now we combine the two bounds above: fix $n \geq 2$, with probability at least $1-\delta$,

$$\|\xi_n(x_n)\|_* + \|\bar{\xi}_{n-1}(x_n)\|_* \leq O\left(\sqrt{\frac{G_2^2}{m_0 n^r}\ln\left(\frac{1}{\delta}\right)} + \sqrt{\left(C_{\mathcal{X}} + \ln\left(\frac{1}{\delta}\right)\right)\frac{G_2^2}{m_0 n^2}\sum_{i=1}^{n}\frac{1}{i^r}}\right)$$

Due to the nature of harmonic series, we consider two scenarios.

1. If $r \in [0,1]$, then the bound can be simplified as

$$\begin{aligned}
&O\left(\sqrt{\frac{G_2^2}{m_0 n^r}\ln\left(\frac{1}{\delta}\right)} + \sqrt{\left(C_{\mathcal{X}} + \ln\left(\frac{1}{\delta}\right)\right)\frac{G_2^2}{m_0 n^2}\sum_{i=1}^{n}\frac{1}{i^r}}\right) \\
&= O\left(\sqrt{\frac{G_2^2}{m_0 n^r}\ln\left(\frac{1}{\delta}\right)} + \sqrt{\left(C_{\mathcal{X}} + \ln\left(\frac{1}{\delta}\right)\right)\frac{G_2^2 n^{1-r}}{m_0 n^2}}\right) \\
&= O\left(G_2\sqrt{\frac{\ln(1/\delta)}{m_0 n^r}} + G_2\sqrt{\frac{C_{\mathcal{X}}}{m_0 n^{1+r}}}\right)
\end{aligned}$$

2. If $r > 1$, then the bound can be simplified as

$$
\begin{aligned}
&O\left(\sqrt{\frac{G_2^2}{m_0 n^r}\ln\left(\frac{1}{\delta}\right)} + \sqrt{\left(C_{\mathcal{X}} + \ln\left(\frac{1}{\delta}\right)\right)\frac{G_2^2}{m_0 n^2}\sum_{i=1}^{n}\frac{1}{i^r}}\right) \\
&= O\left(\sqrt{\frac{G_2^2}{m_0 n^r}\ln\left(\frac{1}{\delta}\right)} + \sqrt{\left(C_{\mathcal{X}} + \ln\left(\frac{1}{\delta}\right)\right)\frac{G_2^2}{m_0 n^2}}\right) \\
&= O\left(G_2\sqrt{\frac{\ln(1/\delta)}{m_0 n^{\min\{r,2\}}}}\right) + O\left(G_2\sqrt{\frac{C_{\mathcal{X}}}{m_0 n^2}}\right)
\end{aligned}
$$

Therefore, we conclude for $r \geq 0$,

$$
\|\xi_n(x_n)\|_* + \|\bar{\xi}_{n-1}(x_n)\|_* = O\left(\sqrt{\frac{G_2^2\ln(1/\delta)}{m_0 n^{\min\{r,2\}}}} + \sqrt{\frac{G_2^2 C_{\mathcal{X}}}{m_0 n^{1+\min\{r,1\}}}}\right)
$$

Combining this inequality with Lemma 6.B.5 gives the final statement. ■

Bound on S_n

Now we use Lemma 6.B.6 to refine Proposition 6.5.1 for stochastic problems.

Proposition 6.B.1. *Under the assumptions Proposition 6.5.1, suppose $m_n = m_0 n^r$. For a fixed $n \geq 2$, the following holds with probability at least $1 - \delta$.*

$$
S_n \leq \tilde{O}\left(\frac{G_2}{\alpha\sqrt{m_0}}\left(\frac{\sqrt{\ln(1/\delta)}}{n^{\min\{r/2,1,1-\theta\}}} + \frac{\sqrt{C_{\mathcal{X}}}}{n^{\min\{(1+r)/2,1,1-\theta\}}}\right)\right)
$$

Proof. The proof is similar to that of Proposition 6.5.1, but we use the results from Lemma 6.B.6. Note Lemma 6.B.6 holds for a particular n. Here need the bound to apply for all $n = 1 \ldots N$ so we can apply the bound for each S_n. This will add an additional $\sqrt{\ln N}$ factor to the bounds in Lemma 6.B.6.

First, we recall that

$$S_{n+1} \leq \left(1 - \frac{1}{n}\right) S_n + \|x_{n+1} - x_n\|$$

By Lemma 6.B.6, let $c_1 = \frac{G_2\sqrt{\ln(1/\delta)}}{n\alpha\sqrt{m_0}}$ and $c_2 = \frac{G_2\sqrt{C_{\mathcal{X}}}}{n\alpha\sqrt{m_0}}$, and it holds that

$$\begin{aligned}\|x_{n+1} - x_n\| &\leq \frac{\theta S_n}{n} + O\left(\frac{G_2}{n\alpha\sqrt{m_0}}\left(\sqrt{\frac{\ln(1/\delta)}{n^{\min\{r,2\}}}} + \sqrt{\frac{C_{\mathcal{X}}}{n^{1+\min\{r,1\}}}}\right)\right) \\ &= \frac{\theta S_n}{n} + O(\frac{c_1}{n^{1+\min\{r,2\}/2}} + \frac{c_2}{n^{3/2+\min\{r,1\}/2}})\end{aligned}$$

which implies

$$S_{n+1} \leq \left(1 - \frac{1}{n}\right) S_n + \|x_{n+1} - x_n\| \leq \left(1 - \frac{1-\theta}{n}\right) S_n + O(\frac{c_1}{n^{1+\min\{r,2\}/2}} + \frac{c_2}{n^{3/2+\min\{r,1\}/2}}).$$

Recall

Lemma 6.A.3. *(Lan, 2013, Lemma 1) Let $\gamma_k \in (0,1)$, $k = 1, 2, \ldots$ be given. If the sequence $\{\Delta_k\}_{k\geq 0}$ satisfies*

$$\Lambda_{k+1} \leq (1 - \gamma_k)\Lambda_k + B_k,$$

then

$$\Lambda_k \leq \Gamma_k + \Gamma_k \sum_{i=1}^{k} \frac{B_i}{\Gamma_{i+1}}$$

where $\Gamma_1 = \Lambda_1$ and $\Gamma_{k+1} = (1 - \gamma_k)\Gamma_k$.

From Proposition 6.5.1, we know the unperturbed dynamics is bounded by $e^{1-\theta}n^{\theta-1}S_2$ (and can be shown in $\Theta(n^{\theta-1})$ as in the proof of Theorem 6.5.3). To consider the effect of the perturbations, due to linearity we can treat each perturbation separately and combine the results by superposition. Suppose a particular perturbation is of the form $O(\frac{C_2}{n^{1+s}})$ for

some C_2 and $s > 0$. By Lemma 6.A.3, suppose $\theta + s < 1$,

$$S_n \leq O(n^{\theta-1}) + O\left(n^{\theta-1}\sum_{k=1}^{n} k^{1-\theta}\frac{C_2}{k^{1+s}}\right) \leq O(n^{\theta-1}) + O\left(C_2 n^{\theta-1} n^{1-s-\theta}\right)$$
$$= O(n^{\theta-1}) + O\left(C_2 n^{-s}\right)$$

For $\theta - s = 1$, $S_n \leq O(n^{\theta-1}) + O(C_2 n^{\theta-1}\ln(n))$; for $\theta + s > 1$, $S_n \leq O(n^{\theta-1}) + O(C_2 n^{\theta-1})$. Therefore, we can conclude $S_n \leq C_1 n^{\theta-1} + \tilde{O}(C_2 n^{-\min\{s,1-\theta\}})$, where the constant $C_1 = e^{1-\theta}S_2$. Finally, using $S_2 \leq \frac{G_2}{\alpha}$ and setting C_2 as c_1 or c_2 gives the final result

$$S_n \leq \tilde{O}\left(\frac{G_2}{\alpha\sqrt{m_0}}\left(\frac{\sqrt{\ln(1/\delta)}}{n^{\min\{r/2,1,1-\theta\}}} + \frac{\sqrt{C_{\mathcal{X}}}}{n^{\min\{(1+r)/2,1,1-\theta\}}}\right)\right)$$

■

Performance Guarantee

Given Proposition 6.B.1, now we can prove the performance of the last iterate.

Theorem 6.5.4. *In addition to Assumptions 6.5.3 and 6.5.4, assume $f(x;s)$ is α-strongly convex in x and $\|\nabla f(x;s)\|_* < G_2$ almost surely. Let $\theta = \frac{\beta}{\alpha}$ and suppose $m_n = m_0 n^r$ for some $r \geq 0$. For all $N > 0$, with probability at least $1-\delta$,*

$$F(x_N, x_N) \leq \tilde{\epsilon}_{\Pi,\pi^\star} + \tilde{O}\left(\frac{\theta^2}{c}\frac{\ln(1/\delta) + C_{\mathcal{X}}}{N^{\min\{r,2,2-2\theta\}}}\right) + \tilde{O}\left(\frac{\ln(1/\delta) + C_{\mathcal{X}}}{cN^{\min\{2,1+r\}}}\right)$$

where $c = \frac{\alpha}{G_2^2 m_0}$ and $C_{\mathcal{X}}$ is a constant[11] *of the complexity of* Π.

Proof. The proof is similar to the proof of Theorem 6.5.2. Let $x_n^* \coloneqq \arg\min_{x\in\mathcal{X}} l_n(x)$.

[11]The constant $C_{\mathcal{X}}$ can be thought as $\ln|\mathcal{X}|$, where $|\mathcal{X}|$ measures the size of $\mathcal{X}$ in e.g. Rademacher complexity or covering number (Mohri, Rostamizadeh, and Talwalkar, 2012). For example, $\ln|\mathcal{X}|$ can be linear in $\dim\mathcal{X}$.

Then

$$
\begin{aligned}
& l_n(x_n) - \min_{x \in \mathcal{X}} l_n(x) \\
& \leq \langle \nabla l_n(x_n), x_n - x_n^* \rangle - \frac{\alpha}{2} \|x_n - x_n^*\|^2 \\
& \leq \left\langle \nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n), x_n - x_n^* \right\rangle + \left\langle \nabla \bar{l}_{n-1}(x_n) - \nabla \bar{g}_{n-1}(x_n), x_n - x_n^* \right\rangle - \frac{\alpha}{2} \|x_n - x_n^*\|^2 \\
& \leq \|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_* \|x_n - x_n^*\| + \|\nabla \bar{g}_{n-1}(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_* \|x_n - x_n^*\| - \frac{\alpha}{2} \|x_n - x_n^*\|^2 \\
& \leq \frac{(\|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_* + \|\nabla \bar{g}_{n-1}(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_*)^2}{2\alpha} \\
& \leq \frac{\|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_*^2 + \|\nabla \bar{g}_{n-1}(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_*^2}{\alpha}
\end{aligned}
$$

where the second inequality is due to $x_n = \arg\min_{x \in \mathcal{X}} \bar{g}_{n-1}(x)$. To bound the first term, recall the fact that $\|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_* < \beta S_n$ and recall by Proposition 6.B.1 that

$$
S_n \leq \tilde{O}\left(\frac{G_2}{\alpha \sqrt{m_0}} \left(\frac{\sqrt{\ln(1/\delta)}}{n^{\min\{r/2,1,1-\theta\}}} + \frac{\sqrt{C_{\mathcal{X}}}}{n^{\min\{(1+r)/2,1,1-\theta\}}} \right) \right)
$$

For the second term, we use the proof in Lemma 6.B.6 with an additional $\ln(N)$ factor, i.e.

$$
\|\nabla \bar{g}_{n-1}(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_* = \tilde{O}\left(\frac{G_2}{\sqrt{m_0}} \sqrt{\frac{\ln(1/\delta) + C_{\mathcal{X}}}{n^{1+\min\{r,1\}}}} \right)
$$

Let $c = \frac{\alpha m_0}{G_2^2}$. Therefore, combining all the results, we have the following with probability

at least $1-\delta$:

$$
\begin{aligned}
& l_n(x_n) - \min_{x\in\mathcal{X}} l_n(x) \\
& \leq \frac{\|\nabla l_n(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_*^2 + \|\nabla \bar{g}_{n-1}(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_*^2}{\alpha} \\
& \leq \frac{\beta^2 S_n^2}{\alpha} + \frac{\|\nabla \bar{g}_{n-1}(x_n) - \nabla \bar{l}_{n-1}(x_n)\|_*^2}{\alpha} \\
& \leq \tilde{O}\left(\frac{\theta^2 G_2^2}{\alpha m_0} \frac{\ln(1/\delta)}{n^{2\min\{r/2,1,1-\theta\}}}\right) + \tilde{O}\left(\frac{\theta^2 G_2^2}{\alpha m_0} \frac{C_{\mathcal{X}}}{n^{2\min\{(r+1)/2,1,1-\theta\}}}\right) + \tilde{O}\left(\frac{G_2^2}{\alpha m_0} \frac{\ln(1/\delta)+C_{\mathcal{X}}}{n^{1+\min\{r,1\}}}\right) \\
& = \tilde{O}\left(\frac{\theta^2}{c} \frac{\ln(1/\delta)}{n^{2\min\{r/2,1,1-\theta\}}}\right) + \tilde{O}\left(\frac{\theta^2}{c} \frac{C_{\mathcal{X}}}{n^{2\min\{(r+1)/2,1,1-\theta\}}}\right) + \tilde{O}\left(\frac{1}{c} \frac{\ln(1/\delta)+C_{\mathcal{X}}}{n^{1+\min\{r,1\}}}\right) \\
& \leq \tilde{O}\left(\frac{\theta^2}{c} \frac{\ln(1/\delta)+C_{\mathcal{X}}}{n^{2\min\{r/2,1,1-\theta\}}}\right) + \tilde{O}\left(\frac{\ln(1/\delta)+C_{\mathcal{X}}}{cn^{1+\min\{r,1\}}}\right)
\end{aligned}
$$

Note the last inequality is unnecessary and is used to simplify the result. It can be seen that the upper bound originally has a weaker dependency on $C_{\mathcal{X}}$.

■

6.C AGGREVATTE with Function Approximations

Here we give a sketch of applying the techniques used in Theorem 6.5.4 to problems where a function approximator is used to learn $f(\cdot; s)$, as in the case considered by Ross, Gordon, and Bagnell (2011) for learning the Q-function.

We consider a meta learning scenario where a linear function approximator $\hat{f}(x, s) = \phi(x, s)^\top w$ is used to approximate $f(x; s)$. We assume $\phi(x, s)^\top w$ satisfies Assumption 6.5.1 and Assumption 6.5.3 with some appropriate constants.

Now we analyze the case where $\sum_{i=1}^{m_n} \hat{f}(\cdot, s_{n,i})$ is used as the per-round loss in AGGREVATTE. Specifically, in the nth iteration of AGGREVATTE, m_n samples$\{f(x_n; s_{n,k})\}_{k=1}^{m_n}$ are first collected, and then w_n is updated by

$$
w_n = \arg\min_{w\in\mathcal{W}} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(f(x_i; s_{i,j}) - \phi(x_i, s_{i,j})^\top w\right)^2 \tag{6.17}
$$

where $\mathcal{W}$ is the domain of w. Given the new w_n, the policy is updated by

$$x_{n+1} = \arg\min_{x \in \mathcal{X}} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \phi(x_i, s_{i,j})^\top w_n \tag{6.18}$$

To prove the performance, we focus on the inequality used in the proof of performance in Theorem 6.5.4.

$$l_n(x_n) - \min_{x \in \mathcal{X}} l_n(x) \leq \langle \nabla l_n(x_n), x_n - x_n^* \rangle - \frac{\alpha}{2} \|x_n - x_n^*\|^2$$

And we expand the inner product term:

$$\begin{aligned} \langle \nabla l_n(x_n), x_n - x_n^* \rangle = \left\langle \nabla \bar{g}_{n;w_{n-1}}(x_n), x_n - x_n^* \right\rangle + \left\langle \nabla \bar{g}_{n;w_n} - \nabla \bar{g}_{n;w_{n-1}}, x_n - x_n^* \right\rangle \\ + \langle \nabla l_n(x_n) - \nabla \bar{g}_{n;w_n}, x_n - x_n^* \rangle \end{aligned}$$

where $\bar{g}_{n;w_n}$ is the finite-sample approximation using w_n . By (6.18), $x_n = \arg\min_{x \in \mathcal{X}} \bar{g}_{n;w_{n-1}}(x)$, and therefore

$$\langle \nabla l_n(x_n), x_n - x_n^* \rangle \leq \left\langle \nabla \bar{g}_{n;w_n} - \nabla \bar{g}_{n;w_{n-1}}, x_n - x_n^* \right\rangle + \langle \nabla l_n(x_n) - \nabla \bar{g}_{n;w_n}, x_n - x_n^* \rangle$$

In the first term, $\|\nabla \bar{g}_{n;w_n} - \nabla \bar{g}_{n;w_{n-1}}\|_* \leq O(\|w_n - w_{n-1}\|)$. As w_n is updated by another value aggregation algorithm, this term can be further bounded similarly as in Lemma 6.5.1, by assuming a similar condition like Assumption 6.5.3 but on the change of the gradient in the objective function in (6.17). In the second term, $\|\nabla l_n(x_n) - \nabla \bar{g}_{n;w_n}\|_*$ can be bounded by the uniform bound of vector-valued martingale in Lemma 6.B.4. Given these two bounds, it follows that

$$l_n(x_n) - \min_{x \in \mathcal{X}} l_n(x) \leq \frac{\|\nabla \bar{g}_{n;w_n} - \nabla \bar{g}_{n;w_{n-1}}\|_*^2 + \|\nabla l_n(x_n) - \nabla \bar{g}_{n;w_n}\|_*^2}{\alpha}$$

Compared with Theorem 6.5.4, since here additional Lipschitz constant is introduced to

bound the change $\|\nabla \bar{g}_{n;w_n} - \nabla \bar{g}_{n;w_{n-1}}\|_*$, one can expect that the stability constant θ for this meta-learning problem will increase.

6.D Weighted Regularization

Here we discuss the case where $R(x) = F(\pi^\star, x)$ regardless the condition $R(x) \geq 0$.

Corollary 6.D.1. *Let* $\tilde{F}(x, x) = F(x, x) + \lambda F(\pi^\star, x)$. *Suppose* $\forall x \in \mathcal{X}$, $\min_{x \in \mathcal{X}} \tilde{F}(x, x) \leq (1 + \lambda)\tilde{\epsilon}_{\Pi,\pi^\star}$. *Define* $\Delta_N = (1 + \lambda)\frac{(\tilde{\theta} e^{1-\tilde{\theta}} G_2)^2}{2\alpha} N^{2(\tilde{\theta}-1)}$. *Running* AGGREVATTE *with* $\tilde{F}$ *in* (6.14) *as the per-round loss has performance satisfies: for all* $N > 0$,

$$\begin{aligned} F(x_N, x_N) &\leq (1 + \lambda)\tilde{\epsilon}_{\Pi,\pi^\star} - \lambda F(x^*, x_N) + \Delta_N \\ &\leq \Delta_N + \tilde{\epsilon}_{\Pi,\pi^\star} + \lambda G_2 \left(\frac{2\lambda G_2}{\alpha} + \sqrt{\frac{2\Delta_N}{\alpha}} \right) \end{aligned}$$

Proof. The first inequality can be seen by the definition $F(x_N, x_N) = \tilde{F}(x_N, x_N) - \lambda F(x^*, x_N)$ and then by applying Theorem 6.5.2 to $\tilde{F}(x_N, x_N)$.

The second inequality shows that $-F(x^*, x_N)$ cannot be too large. Let $l_*(x) = F(x^*, x)$ and $x_N^* = \arg\min_{x \in \mathcal{X}} l_N(x)$. Then

$$\begin{aligned} l_N(x_N) &= l_N(x_N) + \lambda l_*(x_N) - \lambda l_*(x_N) \\ &\leq \Delta_N - \lambda l_*(x_N) + \min_{x \in \mathcal{X}} l_N(x) + \lambda l_*(x) \\ &\leq \Delta_N + l_N(x_N^*) + \lambda(l_*(x_N^*) - l_*(x_N)) \\ &\leq \Delta_N + l_N(x_N^*) + \lambda G_2 \|x_N^* - x_N\| \end{aligned}$$

where the first inequality is due to Theorem 6.5.2 and the third inequality is due to l_* is G_2-Lipschitz continuous. Further, since l_N is α-strongly convex,

$$\frac{\alpha}{2}\|x_N^* - x_N\|^2 \leq l_N(x_N) - l_N(x_N^*) \leq \Delta_N + \lambda G_2 \|x_N^* - x_N\|$$

which implies

$$\|x_N^* - x_N\| \leq \frac{\lambda G_2 + \sqrt{\lambda^2 G^2 + 2\alpha\Delta_N}}{\alpha} \leq \frac{2\lambda G_2 + \sqrt{2\alpha\Delta_N}}{\alpha}$$

Therefore,

$$\begin{aligned} l_N(x_N) &\leq \Delta_N + l_N(x_N^*) + \lambda G_2 \|x_N^* - x_N\| \\ &\leq \Delta_N + \tilde{\epsilon}_{\Pi,\pi^\star} + \lambda G_2 \left(\frac{2\lambda G_2}{\alpha} + \sqrt{\frac{2\Delta_N}{\alpha}} \right) \end{aligned}$$ ■

Corollary 6.D.1 indicates that when $\pi^\star$ is better than all policies under the distribution of $\pi^\star$ (i.e. $F(x^*, x) \geq 0, \forall x \in \mathcal{X}$), then using AGGREVATTE with the weighted problem such that $\tilde{\theta} < 1$ generates a convergent sequence and then the performance on the last iterate is bounded by $(1 + \lambda)\tilde{\epsilon}_{\Pi,\pi^\star} + \Delta_N$. That is, it only introduces a multiplicative constant on $\tilde{\epsilon}_{\Pi,\pi^\star}$. Therefore, the bias due to regularization can be ignored by choosing a larger policy class. This suggests for applications like DAGGER introducing additional weighted cost $\lambda F(x^*, x)$ (i.e. demonstration samples collected under the expert policy's distribution) does not hurt.

However, in generally, $F(x^*, x_N)$ can be negative, when there is a better policy in Π than $\pi^\star$ in sense of the state distribution $d^{\pi^\star}(s)$ generated by the expert policy $\pi^\star$. Corollary 6.D.1 also shows this additional bias introduced by AGGREVATTE is bounded at most $O(\frac{\lambda^2 G_2^2}{\alpha})$.

CHAPTER 7

ACCELERATING IMITATION LEARNING WITH PREDICTIVE MODELS

7.1 Introduction

As shown in the previous chapters, imitation learning (IL) has recently received attention for its ability to speed up policy learning when solving reinforcement learning problems (RL) (Abbeel and Ng, 2005; Chang et al., 2015; Le et al., 2018; Ross and Bagnell, 2014; Ross, Gordon, and Bagnell, 2011; Sun et al., 2017). Unlike pure RL techniques, which rely on uniformed random exploration to locally improve a policy, IL leverages prior knowledge about a problem in terms of *expert demonstrations*. At a high level, this additional information provides policy learning with an informed search direction toward the expert policy.

The goal of IL is to quickly learn a policy that can perform at least as well as the expert policy. Because the expert policy may be suboptimal with respect to the RL problem of interest, performing IL is often used to provide a good warm start to the RL problem, so that the number of interactions with the environment can be minimized. Sample efficiency is especially critical when learning is deployed in applications like robotics, where every interaction incurs real-world costs.

By reducing IL to an online learning problem, *online IL* (Ross, Gordon, and Bagnell, 2011) provides a framework for convergence analysis and mitigates the covariate shift problem encountered in batch IL (Argall et al., 2009; Bojarski et al., 2017). In particular, under proper assumptions, the performance of a policy sequence updated by Follow-the-Leader (FTL) can converge on average to the performance of the expert policy (Ross, Gordon, and Bagnell, 2011). Recently, it was shown that this rate is sufficient to make IL more efficient than solving an RL problem from scratch (Cheng et al., 2018a) (see Chapter 5).

In this chapter, we further accelerate the convergence rate of online IL. Inspired by

the observation we made in Chapter 6 (Cheng and Boots, 2018) that the online learning problem of IL is *not* truly adversarial, we propose two MOdel-Based IL (MOBIL) algorithms, MOBIL-VI and MOBIL-PROX, that can achieve a fast rate of convergence. Under the same assumptions of Ross, Gordon, and Bagnell (2011), these algorithms improve on-average convergence to $O(1/N^2)$, e.g., when a dynamics model is learned online, where N is the number of iterations of policy update.

The improved speed of our algorithms is attributed to using a model oracle to predict the gradient of the next per-round loss in online learning. This model can be realized, e.g., using a simulator based on a (learned) dynamics model, or using past demonstrations. We first conceptually show that this idea can be realized as a variational inequality problem in MOBIL-VI. Next, we propose a practical first-order stochastic algorithm MOBIL-PROX, which alternates between the steps of taking the true gradient and of taking the model gradient. MOBIL-PROX is a generalization of stochastic MIRROR-PROX proposed by Juditsky, Nemirovski, and Tauvel (2011) to the case where the problem is weighted and the vector field is unknown but learned online. In theory, we show that having a *weighting* scheme is pivotal to speeding up convergence, and this generalization is made possible by a new constructive FTL-style regret analysis, which greatly simplifies the original algebraic proof (Juditsky, Nemirovski, and Tauvel, 2011). The performance of MOBIL-PROX is also empirically validated in simulation. This chapter is partly based on our paper published as (Cheng et al., 2019b).

7.2 Preliminaries

7.2.1 Problem Setup: RL and IL

Let $\mathcal{S}$ and $\mathcal{A}$ be the state and the action spaces, respectively. The objective of RL is to search for a stationary policy π inside a policy class Π with good performance. This can be characterized by the stochastic optimization problem with expected cost[1] $\bar{J}(\pi)$ defined

[1] Our definition of $\bar{J}(\pi)$ corresponds to the average accumulated cost in the RL literature.

below:

$$\min_{\pi \in \Pi} \bar{J}(\pi), \quad \bar{J}(\pi) \coloneqq \mathbb{E}_{s \sim d^\pi} \mathbb{E}_{a \sim \pi | s} \left[c(s, a) \right], \tag{7.1}$$

in which $s \in \mathcal{S}$, $a \in \mathcal{A}$, c is the instantaneous cost, d^π is a *average state distribution* induced by executing policy π, and π_s is the distribution of action a given state s of π. The policies here are assumed to be parametric. To make the writing compact, we will abuse the notation π to also denote its parameter, and assume Π is a compact convex subset of parameters in some normed space with norm $\| \cdot \|$.

Recall from Chapter 2 that, based on the abstracted distribution d^π, the formulation in (7.1) subsumes multiple discrete-time RL problems. For example, a γ-discounted infinite-horizon problem can be considered by defining the joint distribution $d^\pi(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t d_t^\pi(s)$, in which $d_t^\pi(s)$ denotes the probability distribution of state s at time t under policy π. Similarly, a T-horizon RL problem can be considered by setting $d^\pi(s) = \frac{1}{T} \sum_{t=0}^{T-1} d_t^\pi(s)$, when we embed time information into the definition of $\mathcal{S}$. Note that while we use the notation $\mathbb{E}_{a \sim \pi | s}$, the policy is allowed to be deterministic; in this case, the notation means evaluation. For notational compactness, we will often omit the random variable inside the expectation (e.g. we shorten (7.1) to $\mathbb{E}_{d^\pi} \mathbb{E}_\pi [c]$). In addition, we denote Q^π as the Q-function[2] with respect to π.

In this chapter, we consider IL, which is an indirect approach to solving the RL problem. We assume there is a black-box oracle $\pi^\star$, called the *expert* policy, from which demonstration $a^* \sim \pi(a^*|s^*)$ can be queried for any state $s \in \mathcal{S}$. To satisfy the querying requirement, usually the expert policy is an algorithm; for example, it can represent a planning algorithm which solves a simplified version of (10.1), or some engineered, hard-coded policy (see e.g. Pan et al., 2017a).

The purpose of incorporating the expert policy into solving (7.1) is to quickly obtain a

[2]For example, in a T-horizon problem, for a state $s \in \mathcal{S}$ at time t, $Q^\pi(s, a) = c(s, a) + \mathbb{E}_{\rho^\pi(s,a)} [\sum_{\tau=t}^{T-1} c_\tau(s_\tau, a_\tau)]$, where $\rho^\pi(s, a)$ denotes the distribution of future trajectory $(s_t, a_t, s_{t+1}, \dots s_{T-1}, a_{T-1})$ conditioned on $s_t = s, a_t = a$.

policy π that has reasonable performance. Toward this end, we consider solving a surrogate problem of (10.1),

$$\min_{\pi \in \Pi} \mathbb{E}_{s \sim d^{\pi}}[D(\pi(\cdot|s)||\pi(\cdot|s))], \tag{7.2}$$

where D is a function that measures the difference between two distributions over actions (e.g. KL divergence; see Chapter 3). Importantly, the objective in (7.2) has the property that $D(\pi^{\star}(\cdot|s)||\pi^{\star}(\cdot|s)) = 0$ for any $s \in \mathcal{S}$ and there is constant $C^{\pi^{\star}} \geq 0$ such that $\forall t \in \mathbb{N}, s \in \mathcal{S}, \pi \in \Pi$, it satisfies $\mathbb{E}_{a \sim \pi|s}[Q^{\pi^{\star}}(s,a)] - \mathbb{E}_{a^{*} \sim \pi|s}[Q^{\pi^{\star}}(s,a^{*})] \leq C^{\pi^{\star}} D(\pi(\cdot|s)||\pi(\cdot|s))$, in which $\mathbb{N}$ denotes the set of natural numbers. For simplicity, we will also write $D(\pi(\cdot|s)||\pi'(\cdot|s)) = D(\pi||\pi')$, for some policies π and π'. Then by the Performance Difference Lemma (Kakade and Langford, 2002), it can be shown that the inequality above implies (see also Chapter 6),

$$\bar{J}(\pi) - \bar{J}(\pi^{\star}) \leq C^{\pi^{\star}} \mathbb{E}_{d^{\pi}}[D(\pi^{\star}||\pi)]. \tag{7.3}$$

Therefore, solving (7.2) can lead to a policy that performs similarly to the expert policy $\pi^{\star}$.

7.2.2 Imitation Learning as Online Learning

The surrogate problem in (7.2) is more structured than the original RL problem in (10.1). In particular, when the distance-like function D is given, and we know that $D(\pi^{\star}||\pi)$ is close to zero when π is close to $\pi^{\star}$. On the contrary, $\mathbb{E}_{a \sim \pi|s}[c(s,a)]$ in (7.1) generally can still be large, even if π is a good policy (since it also depends on the state). This *normalization* property is crucial for the reduction from IL to online learning, as we showed in Chapter 6 (Cheng and Boots, 2018).

The reduction is based on observing that, with the normalization property, the expres-

siveness of the policy class Π can be described with a constant ϵ_Π defined as,

$$\epsilon_\Pi \geq \max_{\{\pi_n \in \Pi\}} \min_{\pi \in \Pi} \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{d^{\pi_n}}[D(\pi^\star || \pi)], \tag{7.4}$$

for all $N \in \mathbb{N}$, which measures the average difference between Π and $\pi^\star$ with respect to D and the state distributions visited by a worst possible policy sequence. Ross, Gordon, and Bagnell (2011) make use of this property and reduce (7.2) into an online learning problem by distinguishing the influence of π on d^π and on $D(\pi^\star||\pi)$ in (7.2). To make this transparent, we define a bivariate function

$$F(\pi, \pi') \coloneqq \mathbb{E}_{d^\pi}[D(\pi^\star || \pi')]. \tag{7.5}$$

Using this bivariate function F, the online learning setup can be described as follows: in round n, the learner applies a policy $\pi_n \in \Pi$ and then the environment reveals a per-round loss

$$l_n(\pi) \coloneqq F(\pi_n, \pi) = \mathbb{E}_{d^{\pi_n}}[D(\pi^\star || \pi)]. \tag{7.6}$$

Ross, Gordon, and Bagnell (2011) show that if the sequence $\{\pi_n\}$ is selected by a *no-regret algorithm*, then it will have good performance in terms of (7.2). For example, DAGGER updates the policy by FTL, $\pi_{n+1} = \arg\min_{\pi \in \Pi} l_{1:n}(\pi)$ and has the following guarantee (cf. (Cheng and Boots, 2018)), where we define the shorthand $l_{1:n} = \sum_{m=1}^{n} l_m$.

Theorem 7.2.1. *Let $\mu_l > 0$. If each l_n is μ_l-strongly convex and $\|\nabla l_n(\pi)\|_* \leq G, \forall \pi \in \Pi$, then* DAGGER *has performance on average satisfying*

$$\frac{1}{N} \sum_{n=1}^{N} \bar{J}(\pi_n) \leq \bar{J}(\pi^\star) + C^{\pi^\star} \left(\frac{G^2}{2\mu_l} \frac{\ln N + 1}{N} + \epsilon_\Pi \right). \tag{7.7}$$

First-order variants of DAGGER based on Follow-the-Regularized-Leader (FTRL) have

also been proposed by Sun et al. (2017) and Cheng et al. (2018a), which have the same performance but only require taking a stochastic gradient step in each iteration without keeping all the previous cost functions (i.e. data) as in the original FTL formulation. The bound in Theorem 7.2.1 also applies to the expected performance of a policy randomly picked out of the sequence $\{\pi_n\}_{n=1}^N$, although it does not necessarily translate into the performance of the last policy π_{N+1}, as we showed in Chapter 6 (Cheng and Boots, 2018).

7.3 Accelerating IL with Predictive Models

The reduction-based approach to solving IL has demonstrated sucess in speeding up policy learning. However, because interactions with the environment are necessary to approximately evaluate the per-round loss, it is interesting to determine if the convergence rate of IL can be further improved. A faster convergence rate will be valuable in real-world applications where data collection is expensive.

We answer this question affirmatively. We show that, by modeling[3] $\nabla_2 F$ the convergence rate of IL can potentially be improved by up to an order, where ∇_2 denotes the derivative to the second argument. The improvement comes through leveraging the fact that the per-round loss l_n defined in (7.6) is not completely unknown or adversarial as it is assumed in the most general online learning setting. Because the *same* function F is used in (7.6) over different rounds, the online component actually comes from the reduction made by Ross, Gordon, and Bagnell (2011), which ignores information about how F changes with the left argument; in other words, it omits the variations of d^π when π changes, as we showed in Chapter 6. Therefore, we argue that the original reduction proposed by Ross, Gordon, and Bagnell (2011), while allowing the use of (7.4) to characterize the performance, loses one critical piece of information present in the original RL problem: *both the system dynamics and the expert are the same across different rounds of online learning.*

[3]We define $\nabla_2 F$ as a vector field $\nabla_2 F : \pi \mapsto \nabla_2 F(\pi, \pi)$

We propose two model-based algorithms (MOBIL-VI and MOBIL-PROX) to accelerate IL. The first algorithm, MOBIL-VI, is conceptual in nature and updates policies by solving variational inequality (VI) problems (Facchinei and Pang, 2007). This algorithm is used to illustrate how modeling $\nabla_2 F$ through a *predictive model* $\nabla_2 \hat{F}$ can help to speed up IL, where $\hat{F}$ is a model bivariate function.[4] The second algorithm, MOBIL-PROX is a first-order method. It alternates between taking stochastic gradients by interacting with the environment and querying the model $\nabla_2 \hat{F}$. We will prove that this simple yet practical approach has the same performance as the conceptual one: when $\nabla_2 \hat{F}$ is learned online and $\nabla_2 F$ is realizable, e.g. both algorithms can converge in $O\left(\frac{1}{N^2}\right)$, in contrast to DAGGER's $O\left(\frac{\ln N}{N}\right)$ convergence. In addition, we show the convergence results of MOBIL under relaxed assumptions, e.g. allowing stochasticity, and provide several examples of constructing predictive models. (See Section 7.A for a summary of notation used in this chapter.)

7.3.1 Performance and Average Regret

Before presenting the two algorithms, we first summarize the core idea of the reduction from IL to online learning in a simple lemma, which builds the foundation of our algorithms (proved in Section 7.B.1).

Lemma 7.3.1. *For arbitrary sequences* $\{\pi_n \in \Pi\}_{n=1}^N$ *and* $\{w_n > 0\}_{n=1}^N$*, it holds that*

$$\mathbb{E}\left[\sum_{n=1}^{N} \frac{w_n \bar{J}(\pi_n)}{w_{1:N}}\right] \leq \bar{J}(\pi^\star) + C^{\pi^\star}\left(\epsilon_\Pi^w + \mathbb{E}\left[\frac{\mathrm{regret}_N^w(\Pi)}{w_{1:N}}\right]\right)$$

where $\tilde{l}_n$ *is an unbiased estimate of* l_n*,* $\mathrm{regret}_N^w(\Pi) \coloneqq \max_{\pi\in\Pi} \sum_{n=1}^N w_n \tilde{l}_n(\pi_n) - w_n \tilde{l}_n(\pi)$*,* ϵ_Π^w *is given in Definition 7.4.1, and the expectation is due to sampling* $\tilde{l}_n$*.*

In other words, the on-average performance convergence of an online IL algorithm is

[4] While we only concern predicting the vector field $\nabla_2 F$, we adopt the notation $\hat{F}$ to better build up the intuition, especially of MOBIL-VI; we will discuss other approximations that are not based on bivariate functions in Section 7.3.3.

determined by the rate of the expected weighted average regret $\mathbb{E}\left[\mathrm{regret}_N^w(\Pi)/w_{1:N}\right]$. For example, in DAGGER, the weighting is uniform and $\mathbb{E}\left[\mathrm{regret}_N^w(\Pi)\right]$ is in $O(\log N)$; by Lemma 7.3.1 this rate directly proves Theorem 7.2.1.

7.3.2 Algorithms

From Lemma 7.3.1, we know that improving the regret bound implies a faster convergence of IL. This leads to the main idea of MOBIL-VI and MOBIL-PROX: to use model information to *approximately* play Be-the-Leader (BTL) (Kalai and Vempala, 2005), i.e. $\pi_{n+1} \approx \arg\min_{\pi\in\Pi} l_{1:n+1}(\pi)$. To understand why playing BTL can minimize the regret, we recall a classical regret bound of online learning.[5]

Lemma 7.3.2 (Strong FTL Lemma (McMahan, 2017))**.** *For any sequence of decisions* $\{x_n \in \mathcal{X}\}$ *and loss functions* $\{\zeta_n\}$, $\mathrm{regret}_N(\mathcal{X}) \le \sum_{n=1}^N \zeta_{1:n}(x_n) - \zeta_{1:n}(x_n^\star)$, *where* $x_n^\star \in \arg\min_{x\in\mathcal{X}} \zeta_{1:n}(x)$, *where* $\mathcal{X}$ *is the decision set.*

Namely, if the decision π_{n+1} made in round n in IL is close to the best decision in round $n+1$ after the new per-round loss l_{n+1} is revealed (which depends on π_{n+1}), then the regret will be small.

The two algorithms are summarized in Algorithm 2, which mainly differ in the policy update rule (line 5). Like DAGGER, they both learn the policy in an interactive manner. In round n, both algorithms execute the current policy π_n in the real environment to collect data to define the per-round loss functions (line 3): $\tilde{l}_n$ is an unbiased estimate of l_n in (7.6) for policy learning, and $\tilde{h}_n$ is an unbiased estimate of the per-round loss h_n for model learning. Given the current per-round losss, the two algorithms then update the model (line 4) and the policy (line 5) using the respective rules. Here we use the set $\hat{\mathcal{F}}$, abstractly, to denote the family of predictive models to estimate $\nabla_2 F$, and h_n is defined as an upper bound of the prediction error. For example, $\hat{\mathcal{F}}$ can be a family of dynamics models that are

[5] We use notation l_n and ζ_n to distinguish general online learning problems from online IL problems.

used to simulate the predicted gradients, and $\tilde{h}_n$ is the empirical loss function used to train the dynamics models (e.g. the KL divergence of prediction).

A Conceptual Algorithm: MOBIL-VI

We first present our conceptual algorithm MOBIL-VI, which is simpler to explain. We assume that l_n and h_n are given, as in Theorem 7.2.1. This assumption will be removed in MOBIL-PROX later. To realize the idea of BTL, in round n, MOBIL-VI uses a newly learned predictive model $\nabla_2 \hat{F}_{n+1}$ to estimate of $\nabla_2 F$ in (7.5) and then updates the policy by solving the VI problem below: finding $\pi_{n+1} \in \Pi$ such that $\forall \pi' \in \Pi$,

$$\langle \Phi_n(\pi_{n+1}), \pi' - \pi_{n+1} \rangle \geq 0, \tag{7.8}$$

where the vector field Φ_n is defined as

$$\Phi_n(\pi) = \sum_{m=1}^{n} w_m \nabla l_m(\pi) + w_{n+1} \nabla_2 \hat{F}_{n+1}(\pi, \pi)$$

Suppose $\nabla_2 \hat{F}_{n+1}$ is the partial derivative of some bivariate function $\hat{F}_{n+1}$. If $w_n = 1$, then the VI problem[6] in (7.8) finds a fixed point π_{n+1} satisfying $\pi_{n+1} = \arg\min_{\pi \in \Pi} l_{1:n}(\pi) + \hat{F}_{n+1}(\pi_{n+1}, \pi)$. That is, if $\hat{F}_{n+1} = F$ exactly, then π_{n+1} plays exactly BTL and by Lemma 10.F.2 the regret is non-positive. In general, we can show that, even with modeling errors, MOBIL-VI can still reach a faster convergence rate such as $O\left(\frac{1}{N^2}\right)$, if a non-uniform weighting scheme is used, the model is updated online, and $\nabla_2 F$ is realizable within $\hat{\mathcal{F}}$. The details will be presented in Section 7.4.2.

[6] Because Π is compact, the VI problem in (7.8) has at least one solution (Facchinei and Pang, 2007). If l_n is strongly convex, the VI problem in line 6 of Algorithm 2 is strongly monotone for large enough n and can be solved e.g. by basic projection method (Facchinei and Pang, 2007). Therefore, for demonstration purpose, we assume the VI problem of MOBIL-VI can be exactly solved.

[7] MOBIL-VI assumes $\tilde{l}_n = l_n$ and $\tilde{h}_n = h_n$

Algorithm 2 MOBIL

Input: π_1, N, p
Output: $\bar{\pi}_N$

1: Set weights $w_n = n^p$ for $n = 1, \dots, N$ and sample integer K with $P(K = n) \propto w_n$
2: **for** $n = 1 \dots K - 1$ **do**
3: Run π_n in the real environment to collect data to define $\tilde{l}_n$ and $\tilde{h}_n$[7]
4: Update the predictive model to $\nabla_2 \hat{F}_{n+1}$; e.g., using FTL $\hat{F}_{n+1} = \arg\min_{\hat{F} \in \hat{\mathcal{F}}} \sum_{m=1}^{n} \frac{w_m}{m} \tilde{h}_m(\hat{F})$
5: Update policy to π_{n+1} by (7.8) (MOBIL-VI) or by (7.9) (MOBIL-PROX)
6: **end for**
7: Set $\bar{\pi}_N = \pi_K$

A Practical Algorithm: MOBIL-PROX

While the previous conceptual algorithm achieves a faster convergence, it requires solving a nontrivial VI problem in each iteration. In addition, it assumes l_n is given as a function and requires keeping all the past data to define $l_{1:n}$. Here we relax these unrealistic assumptions and propose MOBIL-PROX. In round n of MOBIL-PROX, the policy is updated from π_n to π_{n+1} by *taking two gradient steps*:

$$\begin{aligned} \hat{\pi}_{n+1} &= \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} w_m \big(\langle g_m, \pi \rangle + r_m(\pi) \big), \\ \pi_{n+1} &= \arg\min_{\pi \in \Pi} w_{n+1} \langle \hat{g}_{n+1}, \pi \rangle + \sum_{m=1}^{n} w_m \big(\langle g_m, \pi \rangle + r_m(\pi) \big) \end{aligned} \tag{7.9}$$

We define r_n as an $\alpha_n \mu_l$-strongly convex function (with $\alpha_n \in (0, 1]$; we recall μ_l is the strongly convexity modulus of l_n) such that π_n is its global minimum and $r_n(\pi_n) = 0$ (e.g. a Bregman divergence). And we define g_n and $\hat{g}_{n+1}$ as estimates of $\nabla l_n(\pi_n) = \nabla_2 F(\pi_n, \pi_n)$ and $\nabla_2 \hat{F}_{n+1}(\hat{\pi}_{n+1}, \hat{\pi}_{n+1})$, respectively. Here we only require $g_n = \nabla \tilde{l}_n(\pi_n)$ to be unbiased, whereas $\hat{g}_n$ could be a biased estimate of $\nabla_2 \hat{F}_{n+1}(\hat{\pi}_{n+1}, \hat{\pi}_{n+1})$.

MOBIL-PROX treats $\hat{\pi}_{n+1}$, which plays FTL with g_n from the real environment, as a rough estimate of the next policy π_{n+1} and uses it to query an gradient estimate $\hat{g}_{n+1}$ from the model $\nabla_2 \hat{F}_{n+1}$. Therefore, the learner's decision π_{n+1} can approximately play BTL. If we compare the update rule of π_{n+1} and the VI problem in (7.8), we can see that

MOBIL-PROX linearizes the problem and attempts to approximate $\nabla_2 \hat{F}_{n+1}(\pi_{n+1}, \pi_{n+1})$ by $\hat{g}_{n+1}$. While the above approximation is crude, interestingly it is sufficient to speed up the convergence rate to be as fast as MOBIL-VI under mild assumptions, as shown later in Section 7.4.3.

7.3.3 Predictive Models

MOBIL uses $\nabla_2 \hat{F}_{n+1}$ in the update rules (7.8) and (7.9) at round n to predict the unseen gradient at round $n+1$ for speeding up policy learning. Ideally $\hat{F}_{n+1}$ should approximate the unknown bivariate function F so that $\nabla_2 F$ and $\nabla_2 \hat{F}_{n+1}$ are close. This condition can be seen from (7.8) and (7.9), in which MOBIL concerns only $\nabla_2 \hat{F}_{n+1}$ instead of $\hat{F}_{n+1}$ directly. In other words, $\nabla_2 \hat{F}_{n+1}$ is used in MOBIL as a first-order oracle, which leverages all the past information (up to the learner playing π_n in the environment at round n) to predict the future gradient $\nabla_2 F_{n+1}(\pi_{n+1}, \pi_{n+1})$, which depends on the decision π_{n+1} the learner is about to make. Hence, we call it a predictive model.

To make the idea concrete, we provide a few examples of these models. By definition of F in (7.5), one way to construct the predictive model $\nabla_2 \hat{F}_{n+1}$ is through a *simulator* with an (online learned) dynamics model, and define $\nabla_2 \hat{F}_{n+1}$ as the simulated gradient (computed by querying the expert along the simulated trajectories visited by the learner). If the dynamics model is exact, then $\nabla_2 \hat{F}_{n+1} = \nabla_2 F$. Note that a stochastic/biased estimate of $\nabla_2 \hat{F}_{n+1}$ suffices to update the policies in MOBIL-PROX.

Another idea is to construct the predictive model through $\tilde{l}_n$ (the stochastic estimate of l_n) and indirectly define $\hat{F}_{n+1}$ such that $\nabla_2 \hat{F}_{n+1} = \nabla \tilde{l}_n$. This choice is possible, because the learner in IL collects *samples* from the environment, as opposed to, literally, gradients. Specifically, we can define $g_n = \nabla \tilde{l}_n(\pi_n)$ and $\hat{g}_{n+1} = \nabla \tilde{l}_n(\hat{\pi}_{n+1})$ in (7.9). The approximation error of setting $\hat{g}_{n+1} = \nabla \tilde{l}_n(\hat{\pi}_{n+1})$ is determined by the convergence and the stability of the learner's policy. If π_n visits similar states as $\hat{\pi}_{n+1}$, then $\nabla \tilde{l}_n$ can approximate $\nabla_2 F$ well at $\hat{\pi}_{n+1}$. Note that this choice is different from using the previous gradient (i.e. $\hat{g}_{n+1} = g_n$)

in optimistic mirror descent/FTL (Rakhlin and Sridharan, 2012), which would have a larger approximation error due to additional linearization.

Finally, we note that while the concept of predictive models originates from estimating the partial derivatives $\nabla_2 F$, a predictive model does not necessarily have to be in the same form. A parameterized vector-valued function can also be directly learned to approximate $\nabla_2 F$, e.g., using a neural network and the sampled gradients $\{g_n\}$ in a supervised learning fashion.

7.4 Theoretical Analysis

Now we prove that using predictive models in MOBIL can accelerate convergence, when proper conditions are met. Intuitively, MOBIL converges faster than the usual adversarial approach to IL (like DAGGER), when the predictive models have smaller errors than not predicting anything at all (i.e. setting $\hat{g}_{n+1} = 0$). In the following analyses, we will focus on bounding the expected weighted average regret, as it directly translates into the average performance bound by Lemma 7.3.1. We define, for $w_n = n^p$,

$$\mathcal{R}(p) \coloneqq \mathbb{E}\left[\mathrm{regret}_N^w(\Pi)/w_{1:N}\right] \tag{7.10}$$

Note that the results below assume that the predictive models are updated using FTL as outlined in Algorithm 2. This assumption applies, e.g., when a dynamics model is learned online in a simulator-oracle as discussed above. We provide full proofs in Section 7.B and provide a summary of notation in Section 7.A.

7.4.1 Assumptions

We first introduce several assumptions to more precisely characterize the online IL problem.

Predictive models Let $\hat{\mathcal{F}}$ be the class of predictive models. We assume these models are Lipschitz continuous in the following sense.

Assumption 7.4.1. There is $L \in [0, \infty)$ such that $\|\nabla_2 \hat{F}(\pi, \pi) - \nabla_2 \hat{F}(\pi', \pi')\|_* \leq L\|\pi - \pi'\|$, $\forall \hat{F} \in \hat{\mathcal{F}}$ and $\forall \pi, \pi' \in \Pi$.

per-round losss The per-round loss l_n for policy learning is given in (7.6), and we define $h_n(\hat{F})$ as an upper bound of $\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}(\pi_n, \pi_n)\|_*^2$ (see e.g. Section 7.C). We make structural assumptions on $\tilde{l}_n$ and $\tilde{h}_n$, similar to the ones made by Ross, Gordon, and Bagnell (2011) (cf. Theorem 7.2.1).

Assumption 7.4.2. Let $\mu_l, \mu_h > 0$. With probability 1, $\tilde{l}_n$ is μ_l-strongly convex, and $\|\nabla \tilde{l}_n(\pi)\|_* \leq G_l$, $\forall \pi \in \Pi$; $\tilde{h}_n$ is μ_h-strongly convex, and $\|\nabla \tilde{h}_n(\hat{F})\|_* \leq G_h$, $\forall \hat{F} \in \hat{\mathcal{F}}$.

By definition, these properties extend to l_n and h_n. We note they can be relaxed to solely *convexity* and our algorithms still improve the best known convergence rate (see Table 7.1 and Section 7.D).

Table 7.1: Convergence Rate Comparison[8]

	$\tilde{h}_n$ convex	$\tilde{h}_n$ strongly convex	Without model
$\tilde{l}_n$ convex	$O(N^{-3/4})$	$O(N^{-1})$	$O(N^{-1/2})$
$\tilde{l}_n$ strongly convex	$O(N^{-3/2})$	$O(N^{-2})$	$O(N^{-1})$

Expressiveness of hypothesis classes We introduce two constants, ϵ_Π^w and $\epsilon_{\hat{\mathcal{F}}}^w$, to characterize the policy class Π and model class $\hat{\mathcal{F}}$, which generalize the idea of (7.4) to stochastic and general weighting settings. When $\tilde{l}_n = l_n$ and θ_n is constant, Definition 7.4.1 agrees with (7.4). Similarly, we see that if $\pi^\star \in \Pi$ and $F \in \hat{\mathcal{F}}$, then ϵ_Π^w and $\epsilon_{\hat{\mathcal{F}}}^w$ are zero.

[8]The rates here assume $\sigma_{\hat{g}}, \sigma_g, \epsilon_{\hat{\mathcal{F}}}^w = 0$. In general, the rate of MOBIL-PROX becomes the improved rate in the table plus the ordinary rate multiplied by $C = \sigma_g^2 + \sigma_{\hat{g}}^2 + \epsilon_{\hat{\mathcal{F}}}^w$. For example, when $\tilde{f}$ is convex and $\tilde{h}$ is strongly convex, MOBIL-PROX converges in $O(1/N + C/\sqrt{N})$, whereas DAGGER converges in $O(G_l^2/\sqrt{N})$.

Definition 7.4.1. A policy class Π is ϵ_Π^w-*close to* $\pi^\star$, if for all $N \in \mathbb{N}$ and weight sequence $\{\theta_n > 0\}_{n=1}^N$ with $\theta_{1:N} = 1$, $\mathbb{E}\big[\max_{\{\pi_n \in \Pi\}} \min_{\pi \in \Pi} \sum_{n=1}^N \theta_n \tilde{l}_n(\pi)\big] \leq \epsilon_\Pi^w$. Similarly, a model class $\hat{\mathcal{F}}$ is $\epsilon_{\hat{\mathcal{F}}}^w$-*close to* F, if $\mathbb{E}\big[\max_{\{\pi_n \in \Pi\}} \min_{\hat{F} \in \hat{\mathcal{F}}} \sum_{n=1}^N \theta_n \tilde{h}_n(\hat{F})\big] \leq \epsilon_{\hat{\mathcal{F}}}^w$. The expectations above are due to sampling $\tilde{l}_n$ and $\tilde{h}_n$.

7.4.2 Performance of MOBIL-VI

Here we show the performance for MOBIL-VI when there is prediction error in $\nabla_2 \hat{F}_n$. The main idea is to treat MOBIL-VI as online learning with prediction (Rakhlin and Sridharan, 2012) and take $\hat{F}_{n+1}(\pi_{n+1}, \cdot)$ obtained after solving the VI problem (7.8) as an *estimate* of l_{n+1}.

Proposition 7.4.1. *For* MOBIL-VI *with* $p = 0$, $\mathcal{R}(0) \leq \frac{G_l^2}{2\mu_l \mu_h} \frac{1}{N} + \frac{\epsilon_{\hat{\mathcal{F}}}^w}{2\mu_l} \frac{\ln N + 1}{N}$.

By Lemma 7.3.1, this means that if the model class is expressive enough (i.e $\epsilon_{\hat{\mathcal{F}}}^w = 0$), then by adapting the model online with FTL, we can improve the original convergence rate in $O(\ln N/N)$ of Ross, Gordon, and Bagnell (2011) to $O(1/N)$. While removing the $\ln N$ factor does not seem like much, we will show that running MOBIL-VI can improve the convergence rate to $O(1/N^2)$, when a *non-uniform* weighting is adopted.

Theorem 7.4.1. *For* MOBIL-VI *with* $p > 1$, $R(p) \leq C_p \left(\frac{p G_h^2}{2(p-1)\mu_h} \frac{1}{N^2} + \frac{\epsilon_{\hat{\mathcal{F}}}^w}{pN} \right)$, *where* $C_p = \frac{(p+1)^2 e^{p/N}}{2\mu_l}$.

The key is that $\mathrm{regret}_N^w(\Pi)$ can be upper bounded by the regret of the online learning for models, which has per-round loss $\frac{w_n}{n} h_n$. Therefore, if $\epsilon_{\hat{\mathcal{F}}}^w \approx 0$, randomly picking a policy out of $\{\pi_n\}_{n=1}^N$ proportional to weights $\{w_n\}_{n=1}^N$ has expected convergence in $O\left(\frac{1}{N^2}\right)$ if $p > 1$.[9]

7.4.3 Performance of MOBIL-PROX

As MOBIL-PROX uses gradient estimates, we additionally define two constants σ_g and $\sigma_{\hat{g}}$ to characterize the estimation error, where $\sigma_{\hat{g}}$ also entails potential bias.

[9] If $p = 1$, it converges in $O\left(\frac{\ln N}{N^2}\right)$; if $p \in [0, 1)$, it converges in $O\left(\frac{1}{N^{1+p}}\right)$. See Section 7.B.2.

Assumption 7.4.3. $\mathbb{E}[\|g_n - \nabla_2 F(\pi_n, \pi_n)\|_*^2] \le \sigma_g^2$ and $\mathbb{E}[\|\hat{g}_n - \nabla_2 \hat{F}_n(\hat{\pi}_n, \hat{\pi}_n)\|_*^2] \le \sigma_{\hat{g}}^2$

We show this simple first-order algorithm achieves similar performance to MoBIL-VI. Toward this end, we introduce a stronger lemma than Lemma 10.F.2.

Lemma 7.4.1 (Stronger FTL Lemma)**.** *Let* $x_n^\star \in \arg\min_{x\in\mathcal{X}} \zeta_{1:n}(x)$. *For any sequence of decisions* $\{x_n\}$ *and losses* $\{\zeta_n\}$, $\mathrm{regret}_N(\mathcal{X}) = \sum_{n=1}^N \zeta_{1:n}(x_n) - \zeta_{1:n}(x_n^\star) - \Delta_n$, *where* $\Delta_{n+1} := \zeta_{1:n}(x_{n+1}) - \zeta_{1:n}(x_n^\star) \ge 0$.

The additional $-\Delta_n$ term in Lemma 10.F.3 is pivotal to prove the performance of MoBIL-Prox.

Theorem 7.4.2. *For* MoBIL-Prox *with* $p > 1$ *and* $\alpha_n = \alpha \in (0, 1]$, *it satisfies*

$$\mathcal{R}(p) \le \frac{(p+1)^2 e^{\frac{p}{N}}}{\alpha\mu_l}\left(\frac{G_h^2}{\mu_h}\frac{p}{p-1}\frac{1}{N^2} + \frac{2}{p}\frac{\sigma_g^2 + \sigma_{\hat{g}}^2 + \epsilon_{\hat{\mathcal{F}}}^w}{N}\right) + \frac{(p+1)\nu_p}{N^{p+1}},$$

where $\nu_p = O(1)$ *and* $n_{\mathrm{ceil}} = \lceil \frac{2e^{\frac{1}{2}}(p+1)LG_l}{\alpha\mu_l} \rceil$.

Proof sketch. Here we give a proof sketch in big-O notation (see Section 7.B.3 for the details). To bound $\mathcal{R}(p)$, recall the definition $\mathrm{regret}_N^w(\Pi) = \sum_{n=1}^N w_n \tilde{l}_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=1}^N w_n \tilde{l}_n(\pi)$. Now define $\bar{l}_n(\pi) := \langle g_n, \pi\rangle + r_n(\pi)$. Since $\tilde{l}_n$ is μ_l-strongly convex, r_n is $\alpha\mu_l$-strongly convex, and $r(\pi_n) = 0$, we know that $\bar{l}_n$ satisfies that $\tilde{l}_n(\pi_n) - \tilde{l}_n(\pi) \le \bar{l}_n(\pi_n) - \bar{l}_n(\pi), \forall \pi \in \Pi$. This implies $\mathcal{R}(p) \le \mathbb{E}[\mathrm{regret}_N^w p(\Pi)/w_{1:N}]$, where $\mathrm{regret}_N^w p(\Pi) := \sum_{n=1}^N w_n \bar{l}_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=1}^N w_n \bar{l}_n(\pi)$.

The following lemma upper bounds $\mathrm{regret}_N^w p(\Pi)$ by using Stronger FTL lemma (Lemma 10.F.3).

Lemma 7.4.2. $\mathrm{regret}_N^w p(\Pi) \le \frac{p+1}{2\alpha\mu_l} \sum_{n=1}^N n^{p-1} \|g_n - \hat{g}_n\|_*^2 - \frac{\alpha\mu_l}{2(p+1)} \sum_{n=1}^N (n-1)^{p+1} \|\pi_n - \hat{\pi}_n\|^2$.

Since the second term in Lemma 7.4.2 is negative, we just need to upper bound the expectation of the first item. Using the triangle inequality, we bound the model's prediction error of the next per-round loss.

Lemma 7.4.3. $\mathbb{E}[\|g_n - \hat{g}_n\|_*^2] \leq 4(\sigma_g^2 + \sigma_{\hat{g}}^2 + L^2\mathbb{E}[\|\pi_n - \hat{\pi}_n\|^2] + \mathbb{E}[\tilde{h}_n(\hat{F}_n)])$.

With Lemma 7.4.3 and Lemma 7.4.2, it is now clear that $\mathbb{E}[\text{regret}_N^w p(\Pi)] \leq \mathbb{E}[\sum_{n=1}^N \rho_n \|\pi_n - \hat{\pi}_n\|^2] + O(N^p)(\sigma_g^2 + \sigma_{\hat{g}}^2) + O(\mathbb{E}[\sum_{n=1}^N n^{p-1}\tilde{h}_n(\hat{F}_n)])$, where $\rho_n = O(n^{p-1} - n^{p+1})$. When n is large enough, $\rho_n \leq 0$, and hence the first term is $O(1)$. For the third term, because the model is learned online using, e.g., FTL with strongly convex cost $n^{p-1}\tilde{h}_n$ we can show that $\mathbb{E}[\sum_{n=1}^N n^{p-1}\tilde{h}_n(\hat{F}_n)] = O(N^{p-1} + N^p\epsilon_{\hat{\mathcal{F}}}^w)$. Thus, $\mathbb{E}[\text{regret}_N^w p(\Pi)] \leq O(1 + N^{p-1} + (\epsilon_{\hat{\mathcal{F}}}^w + \sigma_g^2 + \sigma_{\hat{g}}^2)N^p)$. Substituting this bound into $\mathcal{R}(p) \leq \mathbb{E}[\text{regret}_N^w p(\Pi)/w_{1:N}]$ and using that the fact $w_{1:N} = \Omega(N^{p+1})$ proves the theorem. ■

The main assumption in Theorem 7.4.2 is that $\nabla_2\hat{F}$ is L-Lipschitz continuous (Assumption 7.4.1). It does not depend on the continuity of $\nabla_2 F$. Therefore, this condition is practical as we are free to choose $\hat{\mathcal{F}}$. Compared with Theorem 7.4.1, Theorem 7.4.2 considers the inexactness of $\tilde{l}_n$ and $\tilde{h}_n$ explicitly; hence the additional term due to σ_g^2 and $\sigma_{\hat{g}}^2$. Under the same assumption of MOBIL-VI that l_n and h_n are directly available, we can actually show that the simple MOBIL-PROX has the same performance as MOBIL-VI, which is a corollary of Theorem 7.4.2.

Corollary 7.4.1. *If* $\tilde{l}_n = l_n$ *and* $\tilde{h}_n = h_n$*, for* MOBIL-PROX *with* $p > 1$*,* $\mathcal{R}(p) \leq O(\frac{1}{N^2} + \frac{\epsilon_{\hat{\mathcal{F}}}^w}{N})$.

The proof of Theorem 7.4.1 and 7.4.2 are based on assuming the predictive models are updated by FTL (see Section 7.C for a specific bound when online learned dynamics models are used as a simulator). However, we note that these results are essentially based on the property that model learning also has no regret; therefore, the FTL update rule (line 4) can be replaced by a no-regret first-order method without changing the result. This would make the algorithm even simpler to implement. The convergence of other types of predictive models (like using the previous cost function discussed in Section 7.3.3) can also be analyzed following the major steps in the proof of Theorem 7.4.2, leading to a performance bound in terms of prediction errors. Finally, it is interesting to note that the

accelerated convergence is made possible when model learning puts more weight on costs in later rounds (because $p > 1$).

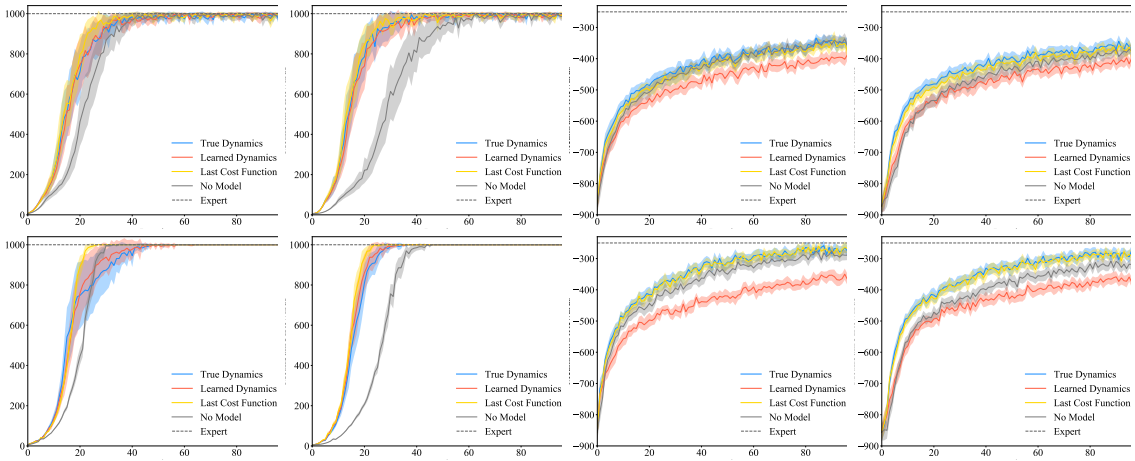
7.4.4 Comparison

We compare the performance of MOBIL in Theorem 7.4.2 with that of DAGGER in Theorem 7.2.1 in terms of the constant on the $\frac{1}{N}$ factor. MOBIL has a constant in $O(\sigma_g^2 + \sigma_{\hat{g}}^2 + \epsilon_{\hat{\mathcal{F}}}^w)$, whereas DAGGER has a constant in $G_l^2 = O(G^2 + \sigma_g^2)$, where we recall G_l and G are upper bounds of $\|\nabla \tilde{l}_n(\pi)\|_*$ and $\|\nabla l_n(\pi)\|_*$, respectively.[10] Therefore, in general, MOBIL-PROX has a better upper bound than DAGGER when the model class is expressive (i.e. $\epsilon_{\hat{\mathcal{F}}} \approx 0$), because $\sigma_{\hat{g}}^2$ (the variance of the sampled gradients) can be made small as we are free to design the model. Note that, however, the improvement of MOBIL may be smaller when the problem is noisy, such that the large σ_g^2 becomes the dominant term.

An interesting property that arises from Theorems 7.4.1 and 7.4.2 is that the convergence of MOBIL is not biased by using an imperfect model (i.e. $\epsilon_{\hat{\mathcal{F}}}^w > 0$). This is shown in the term $\epsilon_{\hat{\mathcal{F}}}^w / N$. In other words, in the worst case of using an extremely wrong predictive model, MOBIL would just converge more slowly but still to the performance of the expert policy.

MOBIL-PROX is closely related to stochastic Mirror-Prox (Juditsky, Nemirovski, and Tauvel, 2011; Nemirovski, 2004). In particular, when the exact model is known (i.e. $\nabla_2 \hat{F}_n = \nabla_2 F$) and MOBIL-PROX is set to convex-mode (i.e. $r_n = 0$ for $n > 1$, and $w_n = 1/\sqrt{n}$; see Section 7.D), then MOBIL-PROX gives the same update rule as stochastic Mirror-Prox with step size $O(1/\sqrt{n})$ (See Section 7.E for a thorough discussion). Therefore, MOBIL-PROX can be viewed as a generalization of Mirror-Prox: 1) it allows non-uniform weights; and 2) it allows the vector field $\nabla_2 F$ to be estimated online by alternately taking stochastic gradients and predicted gradients. The design of MOBIL-PROX is made possible by our Stronger FTL lemma (Lemma 10.F.3), which greatly simplifies the origi-

[10] Theorem 7.2.1 was stated by assuming $l_n = \tilde{l}_n$. In the stochastic setup here, DAGGER has a similar convergence rate in expectation but with G replaced by G_l.



(a) CartPole $p = 0$ (b) CartPole $p = 2$ (c) Reacher3D $p = 0$ (d) Reacher3D $p = 2$

Figure 7.1: Experimental results of MOBIL-PROX with neural network (1st row) and linear policies (2nd row). The shaded regions represent 0.5 standard deviation

nal algebraic proof in (Juditsky, Nemirovski, and Tauvel, 2011; Nemirovski, 2004). Using Lemma 10.F.3 reveals more closely the interactions between model updates and policy updates. In addition, it more clearly shows the effect of non-uniform weighting, which is essential to achieving $O(\frac{1}{N^2})$ convergence. To the best of our knowledge, even the analysis of the original (stochastic) Mirror-Prox from the FTL perspective is new.

7.5 Experiments

We experimented with MOBIL-PROX in simulation to study how weights $w_n = n^p$ and the choice of model oracles affect the learning. We used two weight schedules: $p = 0$ as baseline, and $p = 2$ suggested by Theorem 7.4.2. And we considered several predictive models: (a) a simulator with the true dynamics (b) a simulator with online-learned dynamics (c) the last cost function (i.e. $\hat{g}_{n+1} = \nabla \tilde{l}_n(\hat{\pi}_{n+1})$ (d) no model (i.e. $\hat{g}_{n+1} = 0$; in this case MOBIL-PROX reduces to the first-order version of DAGGER (Cheng et al., 2018a), which is considered as a baseline here).

7.5.1 Setup and Results

Two robot control tasks (CartPole and Reacher3D) powered by the DART physics engine (Lee et al., 2018b) were used as the task environments. The learner was either a linear policy or a small neural network. For each IL problem, an expert policy that shares the same architecture as the learner was used, which was trained using policy gradients. While sharing the same architecture is not required in IL, here we adopted this constraint to remove the bias due to the mismatch between policy class and the expert policy to clarify the experimental results. For MoBIL-Prox, we set $r_n(\pi) = \frac{\mu_l \alpha_n}{2}\|\pi - \pi_n\|^2$ and set α_n such that $\sum w_n \alpha_n \mu_l = (1 + cn^{p+1/2})/\eta_n$, where $c = 0.1$ and η_n was adaptive to the norm of the prediction error. This leads to an effective learning rate $\eta_n w^p/(1 + cn^{p+1/2})$ which is optimal in the convex setting (cf. Table 7.1). For the dynamics model, we used a neural network and trained it using FTL. The results reported are averaged over 24 (CartPole) and 12 (Reacher3D) seeds. Figure 10.4 shows the results of MoBIL-Prox. While the use of neural network policies violates the convexity assumptions in the analysis, it is interesting to see how MoBIL-Prox performs in this more practical setting. We include the experiment details in Section 7.F for completeness.

7.5.2 Discussions

We observe that, when $p = 0$, having model information does not improve the performance much over standard online IL (i.e. no model), as suggested in Proposition 7.4.1. By contrast, when $p = 2$ (as suggested by Theorem 7.4.2), MoBIL-Prox improves the convergence and performs better than not using models.[11] It is interesting to see that this trend also applies to neural network policies.

From Figure 10.4, we can also study how the choice of predictive models affects the convergence. As suggested in Theorem 7.4.2, MoBIL-Prox improves the convergence

[11] We note that the curves between $p = 0$ and $p = 2$ are not directly comparable; we should only compare methods within the same p setting as the optimal step size varies with p. The multiplier on the step size was chosen such that MoBIL-Prox performs similarly in both settings.

only when the model makes non-trivial predictions. If the model is very incorrect, then MOBIL-PROX can be slower. This can be seen from the performance of MOBIL-PROX with online learned dynamics models. In the low-dimensional case of CartPole, the simple neural network predicts the dynamics well, and MOBIL-PROX with the learned dynamics performs similarly as MOBIL-PROX with the true dynamics. However, in the high-dimensional Reacher3D problem, the learned dynamics model generalizes less well, creating a performance gap between MOBIL-PROX using the true dynamics and that using the learned dynamics. We note that MOBIL-PROX would still converge at the end despite the model error. Finally, we find that the performance of MOBIL with the last-cost predictive model is often similar to MOBIL-PROX with the simulated gradients computed through the true dynamics.

7.6 Conclusion

We propose two novel model-based IL algorithms MOBIL-PROX and MOBIL-VI with strong theoretical properties: they are provably up-to-and-order faster than the state-of-the-art IL algorithms and have unbiased performance even when using imperfect predictive models. Although we prove the performance under convexity assumptions, we empirically find that MOBIL-PROX improves the performance even when using neural networks. In general, MOBIL accelerates policy learning when having access to an predictive model that can predict future gradients non-trivially. While the focus of the current paper is theoretical in nature, the design of MOBIL leads to several interesting questions that are important to reliable application of MOBIL-PROX in practice, such as end-to-end learning of predictive models and designing adaptive regularizations for MOBIL-PROX.

7.A Notation

Table 7.2: Summary of Symbols

Symbol	Definition
N	the total number of rounds in online learning
$\bar{J}(\pi)$	the average accumulated cost, $\mathbb{E}_{d^\pi}\mathbb{E}_\pi[c_t]$ of RL in (7.1)
d^π	the generalized stationary state distribution
$D(q\|\|p)$	the difference between distributions p and q
$\pi^\star$	the expert policy
Π	the hypothesis class of policies
π_n	the policy run in the environment at the nth online learning iteration
$\hat{\mathcal{F}}$	the hypothesis class of models (elements denoted as $\hat{F}$)
$\hat{F}_n$	the model used at the $n-1$ iteration to predict the future gradient of the nth iteration
ϵ_Π^w	the policy class complexity (Definition 7.4.1)
$\epsilon_{\hat{\mathcal{F}}}^w$	the model class complexity (Definition 7.4.1)
$F(\pi', \pi)$	the bivariate function $E_{d^{\pi'}}[D(\pi^\star\|\|\pi)]$ in (7.5)
$l_n(\pi)$	$F(\pi_n, \pi)$ in (7.6)
$\tilde{l}_n(\pi)$	an unbiased estimate of $l_n(\pi)$
$h_n(\hat{F})$	an upper bound of $\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}(\pi_n, \pi_n)\|_*^2$
$\tilde{h}_n(\hat{F})$	an unbiased estimate of $h_n(\hat{F})$
μ_l	the modulus of strongly convexity of $\tilde{l}_n$ (Assumption 7.4.2)
G_l	an upper bound of $\|\nabla \tilde{l}_n\|_*$ (Assumption 7.4.2)
G	an upper bound of $\|\nabla l_n\|_*$ (Theorem 7.2.1)
μ_h	modulus of strongly convexity of $\tilde{h}_n$ (Assumption 7.4.2)
G_h	an upper bound of $\|\nabla \tilde{h}_n\|_*$ (Assumption 7.4.2)
L	the Lipschitz constant such that $\|\nabla_2 \hat{F}(\pi, \pi) - \nabla_2 \hat{F}(\pi', \pi')\|_* \leq L\|\pi - \pi'\|$ (Assumption 7.4.1)
$\mathcal{R}(p)$	the expected weighted average regret, $\mathbb{E}\left[\frac{\text{regret}_N^w(\Pi)}{w_{1:N}}\right]$ in (7.10)
regret_N^w	the weighted regret, defined in Lemma 7.3.1
$\{w_n\}$	the sequence of weights used to define regret_N^w; we set $w_n = n^p$

7.B Missing Proofs

7.B.1 Proof of Section 7.3.1

Lemma 7.3.1. *For arbitrary sequences* $\{\pi_n \in \Pi\}_{n=1}^N$ *and* $\{w_n > 0\}_{n=1}^N$*, it holds that*

$$\mathbb{E}\left[\sum_{n=1}^{N} \frac{w_n \bar{J}(\pi_n)}{w_{1:N}}\right] \leq \bar{J}(\pi^\star) + C^{\pi^\star}\left(\epsilon_\Pi^w + \mathbb{E}\left[\frac{\mathrm{regret}_N^w(\Pi)}{w_{1:N}}\right]\right)$$

where $\tilde{l}_n$ *is an unbiased estimate of* l_n*,* $\mathrm{regret}_N^w(\Pi) \coloneqq \max_{\pi\in\Pi} \sum_{n=1}^N w_n \tilde{l}_n(\pi_n) - w_n \tilde{l}_n(\pi)$*,* ϵ_Π^w *is given in Definition 7.4.1, and the expectation is due to sampling* $\tilde{l}_n$*.*

Proof of Lemma 7.3.1. By inequality in (7.3) and definition of l_n,

$$\mathbb{E}\left[\sum_{n=1}^{N} w_n(\bar{J}(\pi_n) - \bar{J}(\pi^\star))\right] \leq C^{\pi^\star}\mathbb{E}\left[\sum_{n=1}^{N} w_n l_n(\pi_n)\right] = C^{\pi^\star}\mathbb{E}\left[\sum_{n=1}^{N} w_n \tilde{l}_n(\pi_n)\right],$$

where the last equality is due to π_n is non-anticipating. This implies that

$$\begin{aligned}\mathbb{E}\left[\sum_{n=1}^{N} w_n \bar{J}(\pi_n)\right] &\leq w_{1:N}\bar{J}(\pi^\star) + C^{\pi^\star}\mathbb{E}\left[\sum_{n=1}^{N} w_n \tilde{l}_n(\pi_n)\right] \\ &= w_{1:N}\bar{J}(\pi^\star) + C^{\pi^\star}\mathbb{E}\left[\min_{\pi\in\Pi}\sum_{n=1}^{N} w_n \tilde{l}_n(\pi) + \mathrm{regret}_N^w(\Pi)\right]\end{aligned}$$

The statement is obtained by dividing both sides by $w_{1:N}$ and by the definition of $\epsilon_{\hat{\mathcal{F}}}^w$. ■

7.B.2 Proof of Section 7.4.2

Theorem 7.4.1. *For* MOBIL-VI *with* $p > 1$*,* $R(p) \leq C_p\left(\frac{pG_h^2}{2(p-1)\mu_h}\frac{1}{N^2} + \frac{\epsilon_{\hat{\mathcal{F}}}^w}{pN}\right)$*, where* $C_p = \frac{(p+1)^2 e^{p/N}}{2\mu_l}$.

Proof. We prove a more general version of Theorem 7.4.1 below. ■

Theorem 7.B.1. *For* MOBIL-VI,

$$\mathcal{R}(p) \leq \begin{cases} \frac{G_h^2}{4\mu_l\mu_h}\frac{p(p+1)^2e^{\frac{p}{N}}}{p-1}\frac{1}{N^2} + \frac{1}{2\mu_l}\frac{(p+1)^2e^{\frac{p}{N}}}{p}\frac{1}{N}\epsilon^w_{\hat{\mathcal{F}}}, & \textit{for } p > 1 \\ \frac{G_h^2}{\mu_l\mu_h}\frac{\ln(N+1)}{N^2} + \frac{2}{\mu_l}\frac{1}{N}\epsilon^w_{\hat{\mathcal{F}}}, & \textit{for } p = 1 \\ \frac{G_h^2}{4\mu_l\mu_h}(p+1)^2\frac{O(1)}{N^{p+1}} + \frac{1}{2\mu_l}\frac{(p+1)^2e^{\frac{p}{N}}}{p}\frac{1}{N^2}\epsilon^w_{\hat{\mathcal{F}}}, & \textit{for } 0 < p < 1 \\ \frac{G_h^2}{2\mu_l\mu_h}\frac{1}{N} + \frac{1}{2\mu_l}\frac{\ln N+1}{N}\epsilon^w_{\hat{\mathcal{F}}}, & \textit{for } p = 0 \end{cases}$$

Proof. The solution π_{n+1} of the VI problem (7.8) satisfies the optimality condition of

$$\pi_{n+1} = \arg\min_{\pi\in\Pi} \sum_{m=1}^{n} w_m l_m(\pi_n) + w_{n+1}\hat{F}_{n+1}(\pi_{n+1}, \pi).$$

Therefore, we can derive the bound of $\mathcal{R}(p)$[12] as

$$\begin{aligned} \mathcal{R}(p) &= \frac{\text{regret}_N^w(\Pi)}{w_{1:N}} \\ &\leq \frac{p+1}{2\mu_l w_{1:N}} \sum_{n=1}^{N} n^{p-1} \|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}_n(\pi_n, \pi_n)\|_*^2 \quad \text{(Lemma 7.G.5)} \\ &\leq \frac{p+1}{2\mu_l w_{1:N}} \sum_{n=1}^{N} n^{p-1} h_n(\pi_n) \quad \text{(Property of } h_n\text{)} \end{aligned} \tag{7.11}$$

Next, we treat $n^{p-1}h_n$ as the per-round loss for an online learning problem, and utilize Lemma 7.G.6 to upper bound the accumulated cost. In particular, we set w_n in Lemma 7.G.6 to n^{p-1} and ζ_n to h_n. Finally, $w_{1:N} = \sum_{n=1}^{N} n^p$ can be lower bounded using Lemma 7.G.1. Hence, for $p > 1$, we have

$$\begin{aligned} \mathcal{R}(p) &\leq \frac{p+1}{2\mu_l}\frac{p+1}{N^{p+1}}\left(\frac{G_h^2}{2\mu_h}\frac{p}{p-1}(N+1)^{p-1} + \frac{1}{p}(N+1)^p\epsilon^w_{\hat{\mathcal{F}}}\right) \\ &= \frac{G_h^2}{4\mu_l\mu_h}\frac{p(p+1)^2}{p-1}\left(\frac{N+1}{N}\right)^{p-1}\frac{1}{N^2} + \frac{1}{2\mu_l}\frac{(p+1)^2}{p}\left(\frac{N+1}{N}\right)^p\frac{1}{N}\epsilon^w_{\hat{\mathcal{F}}} \\ &\leq \frac{G_h^2}{4\mu_l\mu_h}\frac{p(p+1)^2e^{\frac{p}{N}}}{p-1}\frac{1}{N^2} + \frac{1}{2\mu_l}\frac{(p+1)^2e^{\frac{p}{N}}}{p}\frac{1}{N}\epsilon^w_{\hat{\mathcal{F}}}, \end{aligned}$$

[12]The expectation of $\mathcal{R}(p)$ is not required here because MOBIL-VI assumes the problem is deterministic.

where in the last inequality we utilize the fact that $1 + x \leq e^x, \forall x \in \mathbb{R}$. Cases other than $p > 1$ follow from straightforward algebraic simplification. ■

Proposition 7.4.1. *For* MOBIL-VI *with* $p = 0$, $\mathcal{R}(0) \leq \frac{G_l^2}{2\mu_l\mu_h}\frac{1}{N} + \frac{\epsilon_{\hat{\mathcal{F}}}^w}{2\mu_l}\frac{\ln N+1}{N}$.

Proof. Proved in Theorem 7.B.1 by setting $p = 0$. ■

7.B.3 Proof of Section 7.4.3

Lemma 10.F.3 (Stronger FTL Lemma)**.** *Let* $x_n^\star \in \arg\min_{x\in\mathcal{X}} \zeta_{1:n}(x)$. *For any sequence of decisions* $\{x_n\}$ *and losses* $\{\zeta_n\}$, $\mathrm{regret}_N(\mathcal{X}) = \sum_{n=1}^N \zeta_{1:n}(x_n) - \zeta_{1:n}(x_n^\star) - \Delta_n$, *where* $\Delta_{n+1} := \zeta_{1:n}(x_{n+1}) - \zeta_{1:n}(x_n^\star) \geq 0$.

Proof. The proof is based on observing $\zeta_n = \zeta_{1:n} - \zeta_{1:n-1}$ and $\zeta_{1:N}$ as a telescoping sum:

$$\begin{aligned}
\mathrm{regret}_N(\mathcal{X}) &= \sum_{n=1}^N \zeta_n(x_n) - \zeta_{1:N}(x_N^\star) \\
&= \sum_{n=1}^N \left(\zeta_{1:n}(x_n) - \zeta_{1:n-1}(x_n)\right) - \sum_{n=1}^N \left(\zeta_{1:n}(x_n^\star) - \zeta_{1:n-1}(x_{n-1}^\star)\right) \\
&= \sum_{n=1}^N \left(\zeta_{1:n}(x_n) - \zeta_{1:n}(x_n^\star) - \Delta_n\right),
\end{aligned}$$

where for notation simplicity we define $\zeta_{1:0} \equiv 0$. ■

Lemma 7.4.2. $\mathrm{regret}_N^w p(\Pi) \leq \frac{p+1}{2\alpha\mu_l}\sum_{n=1}^N n^{p-1}\|g_n - \hat{g}_n\|_*^2 - \frac{\alpha\mu_l}{2(p+1)}\sum_{n=1}^N (n-1)^{p+1}\|\pi_n - \hat{\pi}_n\|^2$.

Proof. We utilize our new Lemma 10.F.3. First, we bound $\sum_{n=1}^N \zeta_{1:n}(\pi_n) - \zeta_{1:n}(\pi_n^\star)$, where $\pi_n^\star = \arg\min_{\pi\in\Pi} \zeta_{1:n}(\pi)$. We achieve this by Lemma 7.G.4. Let $\zeta_n = w_n\bar{l}_n = w_n(\langle g_n, \pi\rangle + r_n(\pi))$. To use Lemma 7.G.4, we note that because r_n is centered at π_n, π_{n+1} satisfies

$$\begin{aligned}
\pi_{n+1} &= \arg\min_{\pi\in\Pi} \sum_{m=1}^n w_m\bar{l}(\pi) + w_{n+1}\langle\hat{g}_{n+1}, \pi\rangle \\
&= \arg\min_{\pi\in\Pi} \sum_{m=1}^n \underbrace{w_m\bar{l}(\pi)}_{\zeta_n(\pi)} + \underbrace{w_{n+1}\langle\hat{g}_{n+1}, \pi\rangle + w_{n+1}r_{n+1}(\pi_{n+1})}_{v_{n+1}(\pi)}
\end{aligned}$$

Because by definition ζ_n is $w_n\alpha\mu_l$-strongly convex, it follows from Lemma 7.G.4 and Lemma 7.G.1 that

$$\sum_{n=1}^{N} \zeta_{1:n}(\pi_n) - \zeta_{1:n}(\pi_n^\star) \leq \frac{1}{\alpha\mu_l} \sum_{n=1}^{N} \frac{w_n^2}{w_{1:n}} \|\hat{g}_n - g_n\|_*^2 \leq \frac{p+1}{2\alpha\mu_l} \sum_{n=1}^{N} n^{p-1} \|g_n - \hat{g}_n\|_*^2.$$

Next, we bound Δ_{n+1} as follows

$$\begin{aligned}
\Delta_{n+1} &= \zeta_{1:n}(\pi_{n+1}) - \zeta_{1:n}(\pi_n^\star) \\
&\geq \langle \nabla\zeta_{1:n}(\pi_n^\star), \pi_{n+1} - \pi_n^\star \rangle + \frac{\alpha\mu_l w_{1:n}}{2} \|\pi_{n+1} - \pi_n^\star\|^2 && \text{(Strong convexity)} \\
&\geq \frac{\alpha\mu_l w_{1:n}}{2} \|\pi_{n+1} - \pi_n^\star\|^2 && \text{(Optimality condition of } \pi_n^*) \\
&= \frac{\alpha\mu_l w_{1:n}}{2} \|\pi_{n+1} - \hat{\pi}_{n+1}\|^2 && \text{(Definition of } \hat{\pi}_{n+1}) \\
&\geq \frac{\alpha\mu_l n^{p+1}}{2(p+1)} \|\pi_{n+1} - \hat{\pi}_{n+1}\|^2. && \text{(Definition of } w_n \text{ and Lemma 7.G.1)}
\end{aligned}$$

Combining these results proves the bound. ■

Lemma 7.4.3. $\mathbb{E}[\|g_n - \hat{g}_n\|_*^2] \leq 4(\sigma_g^2 + \sigma_{\hat{g}}^2 + L^2\mathbb{E}[\|\pi_n - \hat{\pi}_n\|^2] + \mathbb{E}[\tilde{h}_n(\hat{F}_n)])$.

Proof. By Lemma 7.G.3, we have

$$\begin{aligned}
\mathbb{E}\big[\|g_n - \hat{g}_n\|_*^2\big] \leq 4\Big(&\mathbb{E}\big[\|g_n - \nabla_2 F(\pi_n, \pi_n)\|_*^2\big] + \mathbb{E}\big[\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}_n(\pi_n, \pi_n)\|_*^2\big] + \\
&\mathbb{E}\big[\|\nabla_2 \hat{F}_n(\pi_n, \pi_n) - \nabla_2 \hat{F}_n(\hat{\pi}_n, \hat{\pi}_n)\|_*^2\big] + \mathbb{E}\big[\|\nabla_2 \hat{F}_n(\hat{\pi}_n, \hat{\pi}_n) - \hat{g}_n\|_*^2\big]\Big).
\end{aligned}$$

Because the random quantities are generated in order $\ldots, \pi_n, g_n, \hat{F}_{n+1}, \hat{\pi}_{n+1}, \hat{g}_{n+1}, \pi_{n+1}, g_{n+1} \ldots$, by the variance assumption (Assumption 7.4.3), the first and fourth terms can be bounded by

$$\begin{aligned}
\mathbb{E}\big[\|g_n - \nabla_2 F(\pi_n, \pi_n)\|_*^2\big] &= \mathbb{E}_{\pi_n}\left[\mathbb{E}_{g_n}[\|g_n - \nabla_2 F(\pi_n, \pi_n)\|_*^2 | \pi_n]\right] \leq \sigma_g^2, \\
\mathbb{E}\big[\|\nabla_2 \hat{F}_n(\hat{\pi}_n, \hat{\pi}_n) - \hat{g}_n\|_*^2\big] &= \mathbb{E}_{\hat{F}_n, \hat{\pi}_n}\left[\mathbb{E}_{\hat{g}_n}[\|\nabla_2 \hat{F}_n(\hat{\pi}_n, \hat{\pi}_n) - \hat{g}_n\|_*^2 \big| \hat{\pi}_n, \hat{F}_n]\right] \leq \sigma_{\hat{g}}^2.
\end{aligned}$$

And, for the second term, we have

$$\mathbb{E}\big[\|\nabla_2 F(\pi_n,\pi_n) - \nabla_2 \hat{F}_n(\pi_n,\pi_n)\|_*^2\big] \le \mathbb{E}\big[h_n(\hat{F}_n)\big] = \mathbb{E}\big[\tilde{h}_n(\hat{F}_n)\big]$$

Furthermore, due to the Lipschitz assumption of $\nabla_2 \hat{F}_{n+1}$ (Assumption 7.4.1), the third term is bounded by

$$\mathbb{E}\big[\|\nabla_2 \hat{F}_n(\pi_n,\pi_n) - \nabla_2 \hat{F}_n(\hat{\pi}_n,\hat{\pi}_n)\|_*^2\big] \le L^2 \mathbb{E}\big[\|\pi_n - \hat{\pi}_n\|^2\big].$$

Combing the bounds above, we conclude the lemma.

■

Theorem 7.4.2. *For* MoBIL-Prox *with* $p > 1$ *and* $\alpha_n = \alpha \in (0,1]$*, it satisfies*

$$\mathcal{R}(p) \le \frac{(p+1)^2 e^{\frac{p}{N}}}{\alpha\mu_l}\left(\frac{G_h^2}{\mu_h}\frac{p}{p-1}\frac{1}{N^2} + \frac{2}{p}\frac{\sigma_g^2+\sigma_{\hat{g}}^2+\epsilon_{\hat{\mathcal{F}}}^w}{N}\right) + \frac{(p+1)\nu_p}{N^{p+1}},$$

where $\nu_p = O(1)$ *and* $n_{\text{ceil}} = \lceil \frac{2e^{\frac{1}{2}}(p+1)LG_l}{\alpha\mu_l} \rceil$.

Proof. We prove a more general version of Theorem 7.4.1 below. ■

Theorem 7.B.2. *For* MoBIL-Prox*,*

$$\begin{aligned}
\mathcal{R}(p) &\le \frac{4}{\alpha}\mathcal{R}_{\text{MoBIL-VI}}(p) + \epsilon_{\Pi}^w + \sigma(p)\left(\sigma_g^2+\sigma_{\hat{g}}^2\right) + \frac{(p+1)\nu_p}{N^{p+1}},\\
\sigma(p) &\le \begin{cases} \frac{2}{\alpha\mu_l}\frac{(p+1)^2 e^{\frac{p}{N}}}{p}\frac{1}{N}, & \text{if } p > 0\\ \frac{2}{\alpha\mu_l}\frac{\ln N+1}{N}, & \text{if } p = 0 \end{cases}\\
\nu(p) &= 2e\left(\frac{(p+1)LG_l}{\alpha\mu_l}\right)^2 \sum_{n=2}^{n_{\text{ceil}}} n^{2p-2} - \frac{eG_l^2}{2}\sum_{n=2}^{n_{\text{ceil}}}(n-1)^{p+1}n^{p-1} = O(1),\\
n_{\text{ceil}} &= \lceil \frac{2e^{\frac{1}{2}}(p+1)LG_l}{\alpha\mu_l} \rceil
\end{aligned}$$

where $\mathcal{R}_{\text{MoBIL-VI}}(p)$ *is the upper bound of the average regret* $\mathcal{R}(p)$ *in Theorem 7.B.1, and*

the expectation is due to sampling $\tilde{l}_n$ and $\tilde{h}_n$.

Proof. Recall $\mathcal{R}(p) = \mathbb{E}[\frac{\text{regret}_N^w(\Pi)}{w_{1:N}}]$, where

$$\text{regret}_N^w(\Pi) = \sum_{n=1}^{N} w_n \tilde{l}_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{N} w_n \tilde{l}_n(\pi).$$

Define $\bar{l}_n(\pi) := \langle g_n, \pi \rangle + r_n(\pi)$. Since $\tilde{l}_n$ is μ_l-strongly convex, r_n is $\alpha\mu_l$-strongly convex, and $r(\pi_n) = 0$, $\bar{l}_n$ satisfies

$$\tilde{l}_n(\pi_n) - \tilde{l}_n(\pi) \leq \bar{l}_n(\pi_n) - \bar{l}_n(\pi), \quad \forall \pi \in \Pi.$$

which implies $\mathcal{R}(p) \leq \mathbb{E}[\frac{\text{regret}_N^{w;\text{path}}(\Pi)}{w_{1:N}}]$, where

$$\text{regret}_N^{w;\text{path}}(\Pi) \coloneqq \sum_{n=1}^{N} w_n \bar{l}_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{N} w_n \bar{l}_n(\pi)$$

is regret of an online learning problem with per-round loss $w_n \bar{l}_n$.

upper bounds $\text{regret}_N^{w;\text{path}}(\Pi)$ by using Stronger FTL lemma (Lemma 10.F.3). Since the second term in Lemma 7.4.2 is negative, which is in our favor, we just need to upper bound the expectation of the first item. Using triangular inequality, we proceed to bound $\mathbb{E}\big[\|g_n - \hat{g}_n\|_*^2\big]$, which measures how well we are able to predict the next per-round loss using the model. By substituting the result of Lemma 7.4.3 into Lemma 7.4.2, we see

$$\mathbb{E}\left[\text{regret}_N^{w;\text{path}}(\Pi)\right] \leq \mathbb{E}\Big[\sum_{n=1}^{N} \rho_n \|\pi_n - \hat{\pi}_n\|^2\Big] + \left(\frac{2(p+1)}{\alpha\mu_l} \sum_{n=1}^{N} n^{p-1}\right)\left(\sigma_g^2 + \sigma_{\hat{g}}^2\right) + \frac{2(p+1)}{\alpha\mu_l} \mathbb{E}\Big[\sum_{n=1}^{N} n^{p-1} \tilde{h}_n(\hat{F}_n)\Big] \tag{7.12}$$

where $\rho_n = \frac{2(p+1)L^2}{\alpha\mu_l} n^{p-1} - \frac{\alpha\mu_l}{2(p+1)}(n-1)^{p+1}$. When n is large enough, $\rho_n \leq 0$, and hence

the first term of (7.12) is $O(1)$. To be more precise, $\rho_n \leq 0$ if

$$\begin{aligned}
&\frac{2(p+1)L^2}{\alpha\mu_l} n^{p-1} \leq \frac{\alpha\mu_l}{2(p+1)} (n-1)^{p+1} \\
\iff &(n-1)^2 \geq \left(\frac{2(p+1)LG_l}{\alpha\mu_l}\right)^2 \left(\frac{n}{n-1}\right)^{p-1} \\
\impliedby &(n-1)^2 \geq \left(\frac{2(p+1)LG_l}{\alpha\mu_l}\right)^2 e^{\frac{p-1}{n-1}} \\
\impliedby &(n-1)^2 \geq \left(\frac{2(p+1)LG_l}{\alpha\mu_l}\right)^2 e \qquad \text{(Assume } n \geq p\text{)} \\
\impliedby &n \geq \frac{2e^{\frac{1}{2}}(p+1)LG_l}{\alpha\mu_l} + 1
\end{aligned}$$

Therefore, we just need to bound the first $n_{\text{ceil}} = \lceil \frac{2e^{\frac{1}{2}}(p+1)LG_l}{\alpha\mu_l} \rceil$ terms of $\rho_n \|\pi_n - \hat{\pi}_n\|^2$. Here we use a basic fact of convex analysis in order to bound $\|\pi_n - \hat{\pi}_n\|^2$

Lemma 7.B.1. *Let $\mathcal{X}$ be a compact and convex set and let f, g be convex functions. Suppose $f+g$ is μ-strongly convex. Let $x_1 \in \arg\min_{x\in\mathcal{X}} f(x)$ and $x_2 = \arg\min_{x\in\mathcal{X}} (f(x) + g(x))$. Then $\|x_1 - x_2\| \leq \frac{\|\nabla g(x_1)\|_*}{\mu}$.*

Proof of Lemma 7.B.1. Let $h = f+g$. Because h is μ-strongly convex and $x_2 = \arg\min_{x\in\mathcal{X}} h(x)$

$$\begin{aligned}
\frac{\mu}{2}\|x_1 - x_2\|^2 \leq h(x_1) - h(x_2) &\leq \langle \nabla h(x_1), x_1 - x_2\rangle - \frac{\mu}{2}\|x_1 - x_2\|^2 \\
&\leq \langle \nabla g(x_1), x_1 - x_2\rangle - \frac{\mu}{2}\|x_1 - x_2\|^2
\end{aligned}$$

This implies $\mu\|x_1 - x_2\|^2 \leq \langle \nabla g(x_1), x_1 - x_2\rangle \leq \|\nabla g(x_1)\|_*\|x_1 - x_2\|$. Dividing both sides by $\|x_1 - x_2\|$ concludes the lemma. ■

Utilizing Lemma 7.B.1 and the definitions of π_n and $\hat{\pi}_n$, we have, for $n \geq 2$,

$$\begin{aligned}
\|\pi_n - \hat{\pi}_n\|^2 &\leq \frac{1}{\alpha\mu_l w_{1:n-1}} \|w_n \hat{g}_n\|_*^2 \\
&\leq \frac{(p+1)G_l^2}{\alpha\mu_l} \frac{n^{2p}}{(n-1)^{p+1}} && \text{(Bounded } \hat{g}_n \text{ and Lemma 7.G.1)} \\
&\leq \frac{(p+1)e^{\frac{p+1}{n-1}} G_l^2}{\alpha\mu_l} n^{p-1} && (1 + x \leq e^x) \\
&\leq \frac{e(p+1)G_l^2}{\alpha\mu_l} n^{p-1} && \text{(Assume } n \geq p+2\text{).}
\end{aligned}$$

and therefore, after assuming initialization $\pi_1 = \hat{\pi}_1$, we have the bound

$$\sum_{n=2}^{n_{\text{ceil}}} \rho_n \|\pi_n - \hat{\pi}_n\|^2 \leq 2e \left(\frac{(p+1)LG_l}{\alpha\mu_l} \right)^2 \sum_{n=2}^{n_{\text{ceil}}} n^{2p-2} - \frac{eG_l^2}{2} \sum_{n=2}^{n_{\text{ceil}}} (n-1)^{p+1} n^{p-1} \tag{7.13}$$

For the third term of (7.12), we can tie it back to the bound of $\mathcal{R}(p)$ of MoBIL-VI, which we denote $\mathcal{R}_{\text{MoBIL-VI}}(p)$. More concretely, recall that for MoBIL-VI in (7.11), we have

$$\mathcal{R}(p) \leq \frac{p+1}{2\mu_l w_{1:N}} \sum_{n=1}^{N} n^{p-1} h_n(\pi_n),$$

and we derived the upper bound ($\mathcal{R}_{\text{MoBIL-VI}}(p)$) for the RHS term. By observing that the third term of (7.12) after averaging is

$$\begin{aligned}
\frac{2(p+1)}{\alpha\mu_l w_{1:N}} \mathbb{E}\Big[\sum_{n=1}^{N} n^{p-1} \tilde{h}_n(\hat{F}_n) \Big] &= \mathbb{E}\Big[\frac{4}{\alpha} \left(\frac{p+1}{2\mu_l w_{1:N}} \sum_{n=1}^{N} n^{p-1} \tilde{h}_n(\hat{F}_n) \right) \Big] \\
&\leq \frac{4}{\alpha} \mathbb{E}\Big[\mathcal{R}_{\text{MoBIL-VI}}(p) \Big] \\
&= \frac{4}{\alpha} \mathcal{R}_{\text{MoBIL-VI}}(p).
\end{aligned} \tag{7.14}$$

Dividing (7.12) by $w_{1:N}$, and plugging in (7.13), (7.14), we see

$$\begin{aligned}\mathcal{R}(p) &\le \mathbb{E}[\mathrm{regret}_N^{w;\mathrm{path}}(\Pi)/w_{1:N}] \\ &\le \frac{4}{\alpha}\mathcal{R}_{\text{MoBIL-VI}}(p) + \frac{1}{w_{1:N}}\left(\nu_p + \left(\frac{2(p+1)}{\alpha\mu_l}\sum_{n=1}^{N} n^{p-1}\right)\left(\sigma_g^2 + \sigma_{\hat{g}}^2\right)\right)\end{aligned}$$

where $\nu_p = 2e\left(\frac{(p+1)LG_l}{\alpha\mu_l}\right)^2 \sum_{n=2}^{n_{\text{ceil}}} n^{2p-2} - \frac{eG_l^2}{2}\sum_{n=2}^{n_{\text{ceil}}}(n-1)^{p+1}n^{p-1}$, $n_{\text{ceil}} = \lceil \frac{2e^{\frac{1}{2}}(p+1)LG_l}{\alpha\mu_l} \rceil$.

Finally, we consider the case $p > 1$ as stated in Theorem 7.4.2

$$\begin{aligned}\mathcal{R}(p) &\le \frac{4}{\alpha}\left(\frac{G_h^2}{4\mu_l\mu_h}\frac{p(p+1)^2 e^{\frac{p}{N}}}{p-1}\frac{1}{N^2} + \frac{1}{2\mu_l}\frac{(p+1)^2 e^{\frac{p}{N}}}{p}\frac{1}{N}\epsilon_{\hat{\mathcal{F}}}^w\right) \\ &\quad + \frac{p+1}{N^{p+1}}\left(\nu_p + \left(\frac{2(p+1)}{\alpha\mu_l}\frac{n^p}{p}\right)\left(\sigma_g^2 + \sigma_{\hat{g}}^2\right)\right) \\ &\le \frac{(p+1)^2 e^{\frac{p}{N}}}{\alpha\mu_l}\left(\frac{G_h^2}{\mu_h}\frac{p}{p-1}\frac{1}{N^2} + \frac{2}{p}\frac{\sigma_g^2 + \sigma_{\hat{g}}^2 + \epsilon_{\hat{\mathcal{F}}}^w}{N}\right) + \frac{(p+1)\nu_p}{N^{p+1}},\end{aligned}$$

where $\nu_p = 2e\left(\frac{(p+1)LG_l}{\alpha\mu_l}\right)^2 \left(\frac{(n_{\text{ceil}}+1)^{2p-1}}{2p-1} - 1\right) - \frac{eG_l^2}{2}\frac{(n_{\text{ceil}}-1)^{2p+1}}{2p+1}$, $n_{\text{ceil}} = \lceil \frac{2e^{\frac{1}{2}}(p+1)LG_l}{\alpha\mu_l} \rceil$. ■

7.C Model Learning through Learning Dynamics Models

So far we have stated model learning rather abstractly, which only requires $h_n(\hat{F})$ to be an upper bound of $\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}(\pi_n, \pi_n)\|_*^2$. Now we give a particular example of h_n and $\tilde{h}_n$ when the predictive model is constructed as a simulator with online learned dynamics models. Specifically, we consider learning a transition model $M \in \mathcal{M}$ online that induces a bivariate function $\hat{F}$, where $\mathcal{M}$ is the class of transition models. Let D_{KL} denote the KL divergence and let $d^{\pi_n M}$ be the average state distribution (cf. (10.1)) generated by running policy π_n under transition model M. We define, for $M_n \in \mathcal{M}$, $\hat{F}_n(\pi', \pi) := \mathbb{E}_{d^{\pi'}(M_n)}[D(\pi^\star||\pi)]$. We show the error of $\hat{F}_n$ can be bounded by the KL-divergence error of M_n.

Lemma 7.C.1. *Assume $\nabla D(\pi^\star||\cdot)$ is L_D-Lipschitz continuous with respect to $\|\cdot\|_*$. It*

holds that $\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}_n(\pi_n, \pi_n)\|_*^2 \leq 2^{-1}(L_D \mathrm{Diam}(\mathbb{S}))^2 D_{KL}(d^{\pi_n} || d^{\pi_n}(M_n))$.

Directly minimizing the marginal KL-divergence $D_{KL}(d^{\pi_n}, d^{\pi_n}(M_n))$ is a nonconvex problem and requires backpropagation through time. To make the problem simpler, we further upper bound it in terms of the KL divergence between the true and the modeled *transition probabilities*.

To make the problem concrete, here we consider T-horizon RL problems.

Proposition 7.C.1. *For a T-horizon problem with dynamics P, let M_n be the modeled dynamics. Then $\exists C > 0$ s.t $\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}_n(\pi_n, \pi_n)\|_*^2 \leq \frac{C}{T} \sum_{t=0}^{T-1} (T-t) \mathbb{E}_{d_t^{\pi_n}} \mathbb{E}_\pi \left[D_{KL}(P || M_n) \right]$.*

Therefore, we can simply takes h_n as the upper bound in Proposition 7.C.1, and $\tilde{h}$ as its empirical approximation by sampling state-action transition triples through running policy π_n in the real environment. This construction agrees with the causal relationship assumed in the Section 7.3.2.

7.C.1 Proofs

Lemma 7.C.1. *Assume $\nabla D(\pi^\star || \cdot)$ is L_D-Lipschitz continuous with respect to $\| \cdot \|_*$. It holds that $\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}_n(\pi_n, \pi_n)\|_*^2 \leq 2^{-1}(L_D \mathrm{Diam}(\mathbb{S}))^2 D_{KL}(d^{\pi_n} || d^{\pi_n}(M_n))$.*

Proof. First, we use the definition of dual norm

$$\|\nabla_2 \hat{F}(\pi_n, \pi_n) - \nabla_2 F(\pi_n, \pi_n)\|_* = \max_{\delta : \|\delta\| \leq 1} \left(\mathbb{E}_{d^{\pi_n}} - \mathbb{E}_{d^{\pi_n}(M_n)} \right) \left[\langle \delta, \nabla D(\pi^\star || \pi_n) \rangle \right] \quad (7.15)$$

and then we show that $\langle \delta, \nabla D(\pi^\star || \pi_n) \rangle$ is L_D-Lipschitz continuous: for $\pi, \pi' \in \Pi$,

$$\langle \delta, \nabla D(\pi^\star || \pi) - \nabla D(\pi^\star || \pi') \rangle \leq \|\delta\| \|\nabla D(\pi^\star || \pi) - \nabla D(\pi^\star || \pi')\|_* \leq L_D \|\pi - \pi'\|$$

Note in the above equations ∇ is with respect to $D(\pi^\star || \cdot)$.

Next we bound the right hand side of (7.15) using Wasserstein distance D_W, which is defined as follows (Gibbs and Su, 2002): for two probability distributions p and q defined

on a metric space $D_W(p,q) \coloneqq \sup_{f:\text{Lip}(f(\cdot))\leq 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{x\sim q}[f(x)]$.

Using the property that $\langle \delta, \nabla D(\pi^\star||\pi_n)\rangle$ is L_D-Lipschitz continuous, we can derive

$$\|\nabla_2 \hat{F}(\pi_n,\pi_n) - \nabla_2 F(\pi_n,\pi_n)\|_* \leq L_D D_W(d^{\pi_n}, \hat{d}_{\pi_n}) \leq \frac{L_D \text{Diam}(\mathbb{S})}{\sqrt{2}} \sqrt{D_{KL}(d^{\pi_n}||\hat{d}^n_{\pi_n})}$$

in which the last inequality is due to the relationship between D_{KL} and D_W (Gibbs and Su, 2002). ■

Proposition 7.C.1. *For a T-horizon problem with dynamics P, let M_n be the modeled dynamics. Then $\exists C > 0$ s.t $\|\nabla_2 F(\pi_n,\pi_n) - \nabla_2 \hat{F}_n(\pi_n,\pi_n)\|_*^2 \leq \frac{C}{T}\sum_{t=0}^{T-1}(T-t)\mathbb{E}_{d_t^{\pi_n}}\mathbb{E}_\pi\left[D_{KL}(P||M_n)\right]$.*

Proof. Let $\rho_{\pi,t}$ be the state-action trajectory up to time t generated by running policy π, and let $\hat{\rho}_{\pi,t}$ be that of the dynamics model. To prove the result, we use a simple fact:

Lemma 7.C.2. *Let p and q be two distributions.*

$$KL[p(x,y)||q(x,y)] = KL[p(x)||q(x)] + \mathbb{E}_{p(x)} KL[p(y|x)||q(y|x)]$$

Then the rest follows from Lemma 7.C.1 and the following inequality.

$$\begin{aligned} D_{KL}(d^{\pi_n}||\hat{d}_{\pi_n}) &\leq \frac{1}{T}\sum_{t=0}^{T-1} D_{KL}(\rho_{\pi_n,t}||\hat{\rho}_{\pi_n,t}) \\ &= \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\rho_{\pi_n,t}}\left[\sum_{\tau=0}^{t-1} \ln \frac{p_M(s_{\tau+1}|s_\tau,a_\tau)}{p_{\hat{M}}(s_{\tau+1}|s_\tau,a_\tau)}\right] \\ &= \frac{1}{T}\sum_{t=0}^{T-1}(T-t)\mathbb{E}_{d_t^\pi}\mathbb{E}_\pi\left[D_{KL}(p_M||p_{\hat{M}})\right] \end{aligned}$$ ■

7.D Relaxation of Strong Convexity Assumption

The strong convexity assumption (Assumption 7.4.2) can be relaxed to just convexity. We focus on studying the effect of $\tilde{l}_n$ and/or $\tilde{h}_n$ being just convex on $\mathcal{R}(p)$ in Theorem 7.2.1 and Theorem 7.4.2 in big-O notation. Suggested by Lemma 10.F.2, when strong convexity

is not assumed, additional regularization has to be added in order to keep the stabilization terms $\zeta_{1:n}(x_n) - \zeta_{1:n}(x_n^\star)$ small.

Lemma 7.D.1 (FTRL with prediction)**.** *Let ζ_n be convex with bounded gradient and let $\mathcal{X}$ be a compact set. In round n, let regularization r_n be μ_n-strongly convex for some $\mu_n \geq 0$ such that $r_n(x_n) = 0$ and $x_n \in \arg\min_{\mathcal{X}} r_n(x)$, and let v_{n+1} be a (non)convex function such that $\sum_{m=1}^n w_m (\zeta_n + r_n) + w_{n+1} v_{n+1}$ is convex. Suppose that learner plays Follow-The-Regularized-Leader (FTRL) with prediction, i.e.*

$$x_{n+1} = \arg\min_{x \in \mathcal{X}} \sum_{m=1}^{n} \left(w_m (\zeta_n + r_n) + w_{n+1} v_{n+1} \right)(x)$$

and suppose that $\sum_{m=1}^n w_m \mu_m = \Omega(n^k) > 0$ and $\sum_{m=1}^n w_m r_n(x) \leq O(n^k)$ for all $x \in \mathcal{X}$ and some $k \geq 0$. Then, for $w_n = n^p$,

$$\mathrm{regret}_N^w(\mathcal{X}) = O(N^k) + \sum_{n=1}^{N} O\left(n^{2p-k}\right) \|\nabla \zeta_n(x_n) - \nabla v_n(x_n)\|_*^2$$

Proof. The regret of the online learning problem with *convex* per-round loss $w_n \zeta_n$ can be bounded by the regret of the online learning problem with *strongly convex* per-round loss $w_n (\zeta_n + r_n)$ as follows. Let $x_n^\star \in \arg\min_{x \in \mathcal{X}} \sum_{n=1}^N w_n \zeta_n(x)$.

$$\begin{aligned}
\mathrm{regret}_N^w(\mathcal{X}) &= \sum_{n=1}^{N} w_n \zeta_n(x_n) - \min_{x \in \mathcal{X}} \sum_{n=1}^{N} w_n \zeta_n(x) \\
&= \sum_{n=1}^{N} w_n \left(\zeta_n(x_n) + r_n(x_n)\right) - \sum_{n=1}^{N} w_n \left(\zeta_n(x_n^\star) + r_n(x_n^\star)\right) + \sum_{n=1}^{N} w_n r_n(x_n^\star) \\
&\leq \left(\sum_{n=1}^{N} w_n \left(\zeta_n(x_n) + r_n(x_n)\right) - \min_{x \in \mathcal{X}} \sum_{n=1}^{N} w_n \left(\zeta_n(x) + r_n(x)\right) \right) + O(N^k).
\end{aligned}$$

Since the first term is the regret of the online learning problem with *strongly convex* per-round loss $w_n (\zeta_n + r_n)$, and $x_{n+1} = \arg\min_{\mathcal{X}} \left(\sum_{m=1}^n w_m (\zeta_n + r_n) + w_{n+1} v_{n+1}\right)$, we can bound the first term via Lemma 7.G.5 by setting $w_n = n^p$ and $\sum_{m=1}^n w_m \mu_m = O(n^k)$. ■

The lemma below is a corollary of Lemma 7.D.1.

Lemma 7.D.2 (FTRL)**.** *Under the same condition in Lemma 7.D.1, suppose that learner plays FTRL, i.e.* $x_{n+1} = \arg\min_{\mathcal{X}} \sum_{m=1}^{n} w_m (\zeta_n + r_n)$*. Then, for* $w_n = n^p$ *with* $p > -\frac{1}{2}$*, choose* $\{r_n\}$ *such that* $\sum_{m=1}^{n} w_m \mu_m = \Omega(n^{p+1/2}) > 0$ *and it achieves* $\mathrm{regret}_N^w(\mathcal{X}) = O(N^{p+\frac{1}{2}})$ *and* $\frac{\mathrm{regret}_N^w(\mathcal{X})}{w_{1:N}} = O(N^{-1/2})$.

Proof. Let $\sum_{m=1}^{n} w_m \mu_m = \Theta(n^k) > 0$ for some $k \geq 0$. First, if $2p - k > -1$, then we have

$$\begin{aligned}
\mathrm{regret}_N(\mathcal{X}) &\leq O(N^k) + \sum_{n=1}^{N} O\left(n^{2p-k}\right) \|\nabla \zeta_n(x_n)\|_*^2 && \text{(Lemma 7.D.1)} \\
&\leq O(N^k) + \sum_{n=1}^{N} O\left(n^{2p-k}\right) && (\zeta_n \text{ has bounded gradient}) \\
&\leq O(N^k) + O\left(N^{2p-k+1}\right) && \text{(Lemma 7.G.1)}
\end{aligned}$$

In order to have the best rate, we balance the two terms $O(N^k)$ and $O\left(N^{2p-k+1}\right)$

$$k = 2p - k + 1 \Longrightarrow k = p + \frac{1}{2},$$

That is, $p > -\frac{1}{2}$, because $2p - (p + \frac{1}{2}) > -1$. This setting achieves regret in $O(N^{p+\frac{1}{2}})$. Because $w_{1:N} = O(N^{p+1})$, the average regret is in $O(N^{-\frac{1}{2}})$. ■

With these lemmas, we are ready to derive the upper bounds of $\mathcal{R}(p)$ when either $\tilde{l}_n$ or $\tilde{h}_n$ is just convex, with some minor modification of Algorithm 2. For example, when $\tilde{l}_n$ is only convex, r_n will not be $\alpha\mu_l$ strongly; instead we will concern the strongly convexity of $\sum_{m=1}^{n} w_n r_n$. Similarly, if $\tilde{h}_n$ is only convex, the model cannot be updated by FTL as in line 5 of Algorithm 2; instead it has to be updated by FTRL.

In the following, we will derive the rate for MOBIL-VI (i.e. $\tilde{l}_n = l_n$ and $\tilde{h} = h$) and assume $\epsilon_{\hat{\mathcal{F}}}^w = 0$ for simplicity. The same rate applies to the MOBIL-PROX when there is

no noise. To see this, for example, if $\tilde{l}_n$ is only convex, we can treat r_n as an additional regularization and we can see

$$\mathcal{R}(p) = \mathbb{E}\Big[\frac{\mathrm{regret}_N^w(\Pi)}{w_{1:N}}\Big] \le \frac{1}{w_{1:N}}\mathbb{E}\Big[\underbrace{\sum_{n=1}^N w_n \bar{l}_n(\pi_n) - \min_{\pi\in\Pi}\sum_{n=1}^N w_n \bar{l}_n(\pi)}_{\mathrm{regret}_N^{w;\mathrm{path}}(\Pi)} + \sum_{n=1}^N w_n r_n(\pi_N^\star)\Big]$$

where $\pi_N^\star = \arg\min_{\pi\in\Pi}\sum_{n=1}^N \tilde{l}_n(\pi)$. As in the proof of Theorem 7.4.2, $\mathrm{regret}_N^{w;\mathrm{path}}$ is decomposed into several terms: the $\tilde{h}_n$ part in conjunction with $\sum_{n=1}^N w_n r_n(\pi_N^\star)$ constitute the same $\mathcal{R}(p)$ part for MOBIL-VI, while other terms in $\mathrm{regret}_N^{w;\mathrm{path}}$ are kept the same.

Strongly convex $\tilde{l}_n$ and convex $\tilde{h}_n$ Here we assume $p > \frac{1}{2}$. Under this condition, we have

$$\begin{aligned}
\mathrm{regret}_N^w(\Pi) &= \sum_{n=1}^N O(n^{p-1})\tilde{h}_n(\hat{F}_n) && \text{(Lemma 7.G.5)}\\
&= O\left(N^{p-\frac{1}{2}}\right) && \text{(Lemma 7.D.2)}
\end{aligned}$$

Because $w_{1:N} = \Omega(N^{p+1})$, the average regret $\mathcal{R}(p) = O(N^{-3/2})$.

Convex $\tilde{l}_n$ and strongly convex $\tilde{h}_n$ Here we assume $p > 0$. Suppose $r_{1:n}$ is $\Theta(n^k)$-strongly convex and $2p - k > 0$. Under this condition, we have

$$\begin{aligned}
\mathrm{regret}_N^w(\Pi) &= O(N^k) + \sum_{n=1}^N O\left(n^{2p-k}\right)\tilde{h}_n(\hat{F}_{n+1}) && \text{(Lemma 7.D.1)}\\
&= O\left(N^k\right) + O\left(N^{2p-k}\right). && \text{(Lemma 7.G.6)}
\end{aligned}$$

We balance the two terms and arrive at

$$k = 2p - k \Longrightarrow k = p,$$

which satisfies the condition $2p - k > 0$, if $p > 0$. Because $w_{1:N} = \Omega(N^{p+1})$, the average regret $\mathcal{R}(p) = O(N^{-1})$.

Convex $\tilde{l}_n$ and convex $\tilde{h}_n$ Here we assume $p \geq 0$. Suppose $r_{1:n}$ is $\Theta(n^k)$-strongly convex and $2p - k > -\frac{1}{2}$. Under this condition, we have

$$\begin{aligned}
\operatorname{regret}_N^w(\Pi) &= O(N^k) + \sum_{n=1}^{N} O\left(n^{2p-k}\right) \tilde{h}_n(\hat{F}_{n+1}) && \text{(Lemma 7.D.1)} \\
&= O\left(N^k\right) + O\left(N^{2p-k+\frac{1}{2}}\right) && \text{(Lemma 7.D.1)}
\end{aligned}$$

We balance the two terms and see

$$k = 2p - k + \frac{1}{2} \Longrightarrow k = p + \frac{1}{4},$$

which satisfies the condition $2p-k > -\frac{1}{2}$, if $p \geq 0$. Because $w_{1:N} = \Omega(N^{p+1})$, the average regret $\mathcal{R}(p) = O(N^{-3/4})$.

Convex l_n without model Setting $p = 0$ in Lemma 7.D.2, we have $\operatorname{regret}_N(\Pi) = O(N^{\frac{1}{2}})$.

Therefore, the average regret becomes $O(N^{-\frac{1}{2}})$.

Stochastic problems The above rates assume that there is no noise in the gradient and the model is realizable. If the general case, it should be selected $k = p + 1$ for strongly convex $\tilde{l}_n$ and $k = p + \frac{1}{2}$ for convex $\tilde{l}_n$. The convergence rate will become $O(\frac{\epsilon_{\hat{\mathcal{F}}} + \sigma_g^2 + \sigma_{\hat{g}^2}}{N})$ and $O(\frac{\epsilon_{\hat{\mathcal{F}}} + \sigma_g^2 + \sigma_{\hat{g}^2}}{\sqrt{N}})$, respectively.

7.E Connection with Stochastic Mirror-Prox

In this section, we discuss how MOBIL-PROX generalizes stochastic MIRROR-PROX by Juditsky, Nemirovski, and Tauvel (2011) and Nemirovski (2004) and how the new Stronger

FTL Lemma 10.F.3 provides more constructive and flexible directions to design new algorithms.

7.E.1 Variational Inequality Problems

MIRROR-PROX (Nemirovski, 2004) was first proposed to solve VI problems with monotone operators, which is a unified framework of "convex-like" problems, including convex optimization, convex-concave saddle-point problems, convex multi-player games, and equilibrium problems, etc (see (Facchinei and Pang, 2007) for a tutorial). Here we give the definition of VI problems and review some of its basic properties.

Definition 7.E.1. Let $\mathcal{X}$ be a convex subset in an Euclidean space $\mathcal{E}$ and let $F : \mathcal{X} \to \mathcal{E}$ be an operator, the *VI problem*, denoted as $\mathrm{VI}(\mathcal{X}, F)$, is to find a vector $x^* \in \mathcal{X}$ such that

$$\langle F(x^*), x - x^* \rangle \geq 0, \qquad \forall x \in \mathcal{X}.$$

The set of solutions to this problem is denoted as $\mathrm{SOL}(\mathcal{X}, F)$

It can be shown that, when $\mathcal{X}$ is also compact, then $\mathrm{VI}(\mathcal{X}, F)$ admits at least one solution (Facchinei and Pang, 2007). For example, if $F(x) = \nabla f(x)$ for some function f, then solving $\mathrm{VI}(\mathcal{X}, F)$ is equivalent to finding stationary points.

VI problems are, in general, more difficult than optimization. To make the problem more structured, we will consider the problems equipped with some *general convex* structure, which we define below. When $F(x) = \nabla f(x)$ for some convex function f, the below definitions agree with their convex counterparts.

Definition 7.E.2. An operator $F : \mathcal{X} \to \mathcal{E}$ is called

1. *pseudo-monotone* on $\mathcal{X}$ if for all $x, y \in \mathcal{X}$,

$$\langle F(y), x - y \rangle \geq 0 \implies \langle F(x), x - y \rangle \geq 0$$

2. *monotone* on $\mathcal{X}$ if for all $x, y \in \mathcal{X}$,

$$\langle F(x) - F(y), x - y \rangle \geq 0$$

3. *strictly monotone* on $\mathcal{X}$ if for all $x, y \in \mathcal{X}$,

$$\langle F(x) - F(y), x - y \rangle > 0$$

4. *μ-strongly monotone* on $\mathcal{X}$ if for all $x, y \in \mathcal{X}$,

$$\langle F(x) - F(y), x - y \rangle \geq \mu \|x - y\|^2$$

A VI problem is a special case of general equilibrium problems (Bianchi and Schaible, 1996). Therefore, for a VI problem, we can also define its dual VI problem.

Definition 7.E.3. Given a VI problem $\mathrm{VI}(\mathcal{X}, F)$, the *dual VI problem*, denoted as $\mathrm{DVI}(\mathcal{X}, F)$, is to find a vector $x_D^* \in \mathcal{X}$ such that

$$\langle F(x), x - x_D^* \rangle \geq 0, \qquad \forall x \in \mathcal{X}.$$

The set of solutions to this problem is denoted as $\mathrm{DSOL}(\mathcal{X}, F)$.

The solution sets of the primal and the dual VI problems are connected as given in next proposition, whose proof e.g. can be found in (Konnov and Schaible, 2000).

Proposition 7.E.1.

1. *If F is pseudo-monotone, then* $\mathrm{SOL}(\mathcal{X}, F) \subseteq \mathrm{DSOL}(\mathcal{X}, F)$.

2. *If F is continuous, then* $\mathrm{DSOL}(\mathcal{X}, F) \subseteq \mathrm{SOL}(\mathcal{X}, F)$.

However, unlike primal VI problems, a dual VI problem does not always have a solution even if $\mathcal{X}$ is compact. To guarantee the existence of solution to $\mathrm{DSOL}(\mathcal{X}, F)$ it needs stronger structure, such as pseudo-monotonicity as shown in Proposition 7.E.1. Like solving primal VI problems is related to finding *local* stationary points in optimization, solving dual VI problems is related to finding *global* optima when $F(x) = \nabla f(x)$ for some function f (Komlósi, 1999).

7.E.2 Stochastic Mirror-Prox

Stochastic MIRROR-PROX solves a monotone VI problem by indirectly finding a solution to its dual VI problem using stochastic first-order oracles. This is feasible because of Proposition 7.E.1. The way it works is as follows: given an initial condition $x_1 \in \mathcal{X}$, it initializes $\hat{x}_1 = x_1$; at iteration n, it receives unbiased estimates g_n and $\hat{g}_n$ satisfying $\mathbb{E}[g_n] = F(x_n)$ and $\mathbb{E}[\hat{g}_n] = F(\hat{x}_n)$ and then performs updates

$$\begin{aligned} x_{n+1} &= \mathrm{Prox}_{\hat{x}_n}(\gamma_n \hat{g}_n) \\ \hat{x}_{n+1} &= \mathrm{Prox}_{\hat{x}_n}(\gamma_n g_{n+1}) \end{aligned} \tag{7.16}$$

where $\gamma_n > 0$ is the step size, and the proximal operator Prox is defined as

$$\mathrm{Prox}_y(g) = \arg\min_{x \in \mathcal{X}} \langle g, x \rangle + B_\omega(x||y)$$

and $B_\omega(x||y) = \omega(x) - \omega(y) - \langle \nabla\omega(y), x - y \rangle$ is the Bregman divergence with respect to an α-strongly convex function ω. At the end, stochastic MIRROR-PROX outputs

$$\bar{x}_N = \frac{\sum_{n=1}^{N} \gamma_n x_n}{\gamma_{1:n}}$$

as the final decision.

For stochastic MIRROR-PROX, the accuracy of an candidate solution x is based on the

error

$$\mathrm{ERR}(x) \coloneqq \max_{y \in \mathcal{X}} \langle F(y), x - y \rangle .$$

This choice of error follows from the optimality criterion of the dual VI problem in Definition 7.E.3. That is, $\mathrm{ERR}(x) \leq 0$ if and only if $x \in \mathrm{DSOL}(\mathcal{X}, F)$. From Proposition 7.E.1, we know that if the problem is pseudo-monotone, a dual solution is also a primal solution. Furthermore, we can show an approximate dual solution is also an approximate primal solution.

Let $\Omega^2 = \max_{x,y \in \mathcal{X}} B_\omega(x||y)$. Now we recap the main theorem of (Juditsky, Nemirovski, and Tauvel, 2011).[13]

Theorem 7.E.1. (Juditsky, Nemirovski, and Tauvel, 2011) *Let F be monotone. Assume F is L-Lipschitz continuous, i.e.*

$$\|F(x) - F(y)\|_* \leq L\|x - y\| \qquad \forall x, y \in \mathcal{X}$$

and for all n, the sampled vectors are unbiased and have bounded variance, i.e.

$$\mathbb{E}[g_n] = F(x_n), \qquad \mathbb{E}[\hat{g}_n] = F(\hat{x}_n)$$
$$\mathbb{E}[\|g_n - F(x_n)\|_*^2] \leq \sigma^2, \qquad \mathbb{E}[\|\hat{g}_n - F(\hat{x}_n)\|_*^2] \leq \sigma^2$$

Then for $\gamma_n = \gamma$ with $0 < \gamma_n \leq \frac{\alpha}{\sqrt{3}L}$, it satisfies that

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \frac{2\alpha\Omega^2}{N\gamma} + \frac{7\gamma\sigma^2}{\alpha}$$

[13] Here simplify the condition they made by assuming F is Lipschitz continuous and g_n and $\hat{g}_n$ are unbiased.

In particular, if $\gamma = \min\{\frac{\alpha}{\sqrt{3}L}, \alpha\Omega\sqrt{\frac{2}{7N\sigma^2}}\}$*, then*

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \max\left\{\frac{7}{2}\frac{\Omega^2 L}{\alpha}\frac{1}{N}, \Omega\sqrt{\frac{14\sigma^2}{N}}\right\}$$

If the problem is deterministic, the original bound of Nemirovski (2004) is as follows.

Theorem 7.E.2. (Nemirovski, 2004) *Under the same assumption in Theorem 7.E.1, suppose the problem is deterministic. For* $\gamma \leq \frac{\alpha}{\sqrt{2}L}$*,*

$$\mathrm{ERR}(\bar{x}_N) \leq \sqrt{2}\frac{\Omega^2 L}{\alpha}\frac{1}{N}$$

Unlike the uniform scheme above, a recent analysis by Ho-Nguyen and Kılınç-Karzan (2018) also provides a performance bound the weighted average version of MIRROR-PROX when the problem is deterministic.

Theorem 7.E.3. (Ho-Nguyen and Kılınç-Karzan, 2018) *Under the same assumption in Theorem 7.E.1, suppose the problem is deterministic. Let* $\{w_n \geq 0\}$ *be a sequence of weights and let the step size to be* $\gamma_n = \frac{\alpha}{L}\frac{w_{1:n}}{\max_m w_m}$*.*

$$\mathrm{ERR}(\bar{x}_N) \leq \frac{\Omega^2 L}{\alpha}\frac{\max_n w_n}{w_{1:N}}$$

Theorem 7.E.3 (with $w_n = w$) tightens Theorem 7.E.1 and Theorem 7.E.2 by a constant factor.

7.E.3 Connection with MOBIL-PROX

To relate stochastic MIRROR-PROX and MOBIL-PROX, we first rename the variables in (7.16) by setting $\hat{x}_{n+1} \coloneqq \hat{x}_n$ and $\gamma_{n+1} \coloneqq \gamma_n$

$$\begin{aligned} x_{n+1} &= \mathrm{Prox}_{\hat{x}_n}(\gamma_n \hat{g}_n) \\ \hat{x}_{n+1} &= \mathrm{Prox}_{\hat{x}_n}(\gamma_n g_{n+1}) \end{aligned} \qquad \Longleftrightarrow \qquad \begin{aligned} x_{n+1} &= \mathrm{Prox}_{\hat{x}_{n+1}}(\gamma_{n+1} \hat{g}_{n+1}) \\ \hat{x}_{n+2} &= \mathrm{Prox}_{\hat{x}_{n+1}}(\gamma_{n+1} g_{n+1}) \end{aligned}$$

and then reverse the order of updates and write them as

$$\begin{aligned} \hat{x}_{n+1} &= P_{\hat{x}_n}(\gamma_n g_n) \\ x_{n+1} &= P_{\hat{x}_{n+1}}(\gamma_{n+1} \hat{g}_{n+1}) \end{aligned} \tag{7.17}$$

Now we will show that the update in (7.17) is a special case of (7.9), which we recall below

$$\begin{aligned} \hat{\pi}_{n+1} &= \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} w_m \big(\langle g_m, \pi \rangle + r_m(\pi) \big), \\ \pi_{n+1} &= \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} w_m \big(\langle g_m, \pi \rangle + r_m(\pi) \big) + w_{n+1} \langle \hat{g}_{n+1}, \pi \rangle , \end{aligned} \tag{7.9}$$

That is, we will show that $x_n = \pi_n$ and $\hat{x} = \hat{\pi}_n$ under certain setting.

Proposition 7.E.2. *Suppose* $w_n = \gamma_n$, $\hat{F}_n = F$, $r_1(\pi) = B_\omega(\pi || \pi_1)$ *and* $r_n = 0$ *for* $n > 1$. *If* $\Pi = \mathcal{X}$ *is unconstrained, then* $x_n = \pi_n$ *and* $\hat{x}_n = \hat{\pi}_n$ *as defined in* (7.17) *and* (7.9).

Proof. We prove the assertion by induction. For $n = 1$, it is trivial, since $\pi_1 = \hat{\pi}_1 = x_1 = \hat{x}_1$. Suppose it is true for n. We show it also holds for $n + 1$.

We first show $\hat{x}_{n+1} = \hat{\pi}_{n+1}$. By the optimality condition of $\hat{\pi}_{n+1}$, it holds that

$$\begin{aligned}
0 &= \sum_{m=1}^{n} w_m g_m + \nabla\omega(\hat{\pi}_{n+1}) - \nabla\omega(\pi_1) \\
&= (w_n g_n + \nabla\omega(\hat{\pi}_{n+1}) - \nabla\omega(\hat{\pi}_n)) + \left(\sum_{m=1}^{n-1} w_m g_m + \nabla\omega(\hat{\pi}_n) - \nabla\omega(\pi_1)\right) \\
&= w_n g_n + \nabla\omega(\hat{\pi}_{n+1}) - \nabla\omega(\hat{\pi}_n)
\end{aligned}$$

where the last equality is by the optimality condition of $\hat{\pi}_n$. This is exactly the optimality condition of $\hat{x}_{n+1}$ given in (7.17), as $\hat{x}_n = \hat{\pi}_n$ by induction hypothesis and $w_n = \gamma_n$. Finally, because Prox is single-valued, it implies $\hat{x}_{n+1} = \hat{\pi}_{n+1}$.

Next we show that $\pi_{n+1} = x_{n+1}$. By optimality condition of π_{n+1}, it holds that

$$\begin{aligned}
0 &= w_{n+1}\hat{g}_{n+1} + \sum_{m=1}^{n} w_m g_m + \nabla\omega(\pi_{n+1}) - \nabla\omega(\pi_1) \\
&= (w_{n+1}\hat{g}_{n+1} + \nabla\omega(\pi_{n+1}) - \nabla\omega(\hat{\pi}_{n+1})) + \left(\sum_{m=1}^{n} w_m g_m + \nabla\omega(\hat{\pi}_{n+1}) - \nabla\omega(\pi_1)\right) \\
&= w_{n+1}\hat{g}_{n+1} + \nabla\omega(\pi_{n+1}) - \nabla\omega(\hat{\pi}_{n+1})
\end{aligned}$$

This is the optimality condition also for x_{n+1}, since we have shown that $\hat{\pi}_{n+1} = \hat{x}_{n+1}$. The rest of the argument follows similarly as above. ■

In other words, stochastic MIRROR-PROX is a special case of MOBIL-PROX, when $\hat{F}_n = F$ (i.e. the update of π_n also queries the environment not the simulator) and the regularization is constant. The condition that $\mathcal{X}$ and Π are unconstrained is necessary to establish the exact equivalence between Prox-based updates and FTL-based updates. This is a known property in the previous studies on the equivalence between lazy mirror descent and FTRL (McMahan, 2017). Therefore, when $\hat{F}_n = F$, we can view MOBIL-PROX as a lazy version of MIRROR-PROX. It has been empirical observed the FT(R)L version sometimes empirically perform better than the Prox version (McMahan, 2017).

With the connection established by Proposition 7.E.2, we can use a minor modification of the strategy used in Theorem 7.4.2 to prove the performance of MOBIL-PROX when solving VI problems. To show the simplicity of the FTL-style proof compared with the algebraic proof of Juditsky, Nemirovski, and Tauvel (2011), below we will prove from scratch but only using the new Stronger FTL Lemma (Lemma 10.F.3).

To do so, we introduce a lemma to relate expected regret and $\mathrm{ERR}(\bar{x}_N)$.

Lemma 7.E.1. *Let F be a monotone operator. For any $\{x_n \in \mathcal{X}\}_{n=1}^N$ and $\{w_n \geq 0\}$,*

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \mathbb{E}\left[\max_{x\in\mathcal{X}} \frac{1}{w_{1:N}} \sum_{n=1}^N w_n \left\langle F(x_n), x_n - x\right\rangle\right]$$

where $\bar{x}_N = \frac{\sum_{n=1}^N w_n x_n}{w_{1:n}}$.

Proof. Let $x^\star \in \arg\max_{x\in\mathcal{X}} \left\langle F(x), \bar{x}_N - x\right\rangle$. By monotonicity, for all x_n, $\left\langle F(x^\star), x_n - x^\star\right\rangle \leq \left\langle F(x_n), x_n - x^\star\right\rangle$. and therefore

$$\begin{aligned}\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] &= \mathbb{E}\left[\frac{1}{w_{1:N}} \sum_{n=1}^N w_n \left\langle F(x^\star), x_n - x^\star\right\rangle\right] \\ &\leq \mathbb{E}\left[\frac{1}{w_{1:N}} \sum_{n=1}^N w_n \left\langle F(x_n), x_n - x^\star\right\rangle\right] \leq \mathbb{E}\left[\max_{x\in\mathcal{X}} \frac{1}{w_{1:N}} \sum_{n=1}^N w_n \left\langle F(x_n), x_n - x\right\rangle\right]\end{aligned}$$

■

Theorem 7.E.4. *Under the same assumption as in Theorem 7.E.1. Suppose $w_n = n^p$ and $r_n(x) = \beta_n B_\omega(x||x_n)$, where β_n is selected such that $\sum_{n=1}^N w_n\beta_n = \frac{1}{\eta}n^k$ for some $k \geq 0$ and $\eta > 0$. If $k > p$, then*

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \frac{1}{w_{1:N}}\left(\frac{\alpha\Omega^2}{\eta}N^k + \frac{3\sigma^2\eta}{\alpha}\sum_{n=1}^N n^{2p-k}\right) + \frac{O(1)}{w_{1:N}}$$

Proof. To simplify the notation, define $\zeta_n(x) = w_n(\langle F(x_n), x\rangle + r_n(x))$ and let

$$\begin{aligned}\mathrm{regret}_N^w(\mathcal{X}) &= \sum_{n=1}^N w_n \langle F(x_n), x_n\rangle - \min_{x\in\mathcal{X}} \sum_{n=1}^N w_n \langle F(x_n), x\rangle \\ \mathcal{R}^w(\mathcal{X}) &= \sum_{n=1}^N \zeta_n(x_n) - \min_{x\in\mathcal{X}} \sum_{n=1}^N \zeta_n(x)\end{aligned}$$

By this definition, it holds that

$$\mathrm{regret}_N^w(\mathcal{X}) \le \mathcal{R}^w(\mathcal{X}) + \max_{x\in\mathcal{X}} \sum_{n=1}^N w_n r_n(x)$$

In the following, we bound the two terms in the upper bound above. First, by applying Stronger FTL Lemma (Lemma 10.F.3) with ζ_n and we can show that

$$\begin{aligned}\mathcal{R}^w(\mathcal{X}) &\le \sum_{n=1}^N \zeta_{1:n}(x_n) - \zeta_{1:n}(x_n^\star) - \Delta_n \\ &\le \sum_{n=1}^N \frac{\eta}{2\alpha} n^{2p-k} \|g_n - \hat{g}_n\|_*^2 - \frac{\alpha(n-1)^{k-1}}{2\eta}\|x_n - \hat{x}_n\|^2\end{aligned}$$

where $x_n^\star \coloneqq \arg\max_{x\in\mathcal{X}} \zeta_{1:n}(x)$. Because by Lemma 7.G.3 and Lipschitz continuity of F, it holds

$$\|g_n - \hat{g}_n\|_*^2 \le 3(L^2\|x_n - \hat{x}_n\|^2 + 2\sigma^2) \tag{7.18}$$

Therefore, we can bound

$$\mathcal{R}^w(\mathcal{X}) \le \sum_{n=1}^N \left(\frac{3}{2}\frac{L^2\eta}{\alpha} n^{2p-k} - \frac{\alpha}{2\eta}(n-1)^k\right)\|x_n - \hat{x}_n\|^2 + \frac{3\sigma^2\eta}{\alpha}\sum_{n=1}^N n^{2p-k} \tag{7.19}$$

If $k > p$, then the first term above is $O(1)$ independent of N. On the other hand,

$$\max_{x\in\mathcal{X}} \sum_{n=1}^N w_n r_n(x) \le \frac{\alpha\Omega^2}{\eta} N^k \tag{7.20}$$

Combining the two bounds and Lemma 7.E.1, i.e. $\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \mathbb{E}\left[\frac{\mathrm{regret}_N^w(\mathcal{X})}{w_{1:N}}\right]$ concludes the proof. ■

Deterministic Problems For deterministic problems, we specialize the proof Theorem 7.E.4 gives. We set $k = p = 0$, $x_1 = \arg\min_{x\in\mathcal{X}} \omega(x)$, which removes the 2 factor in (7.20), and modify 3 to 1 in (7.18) (because the problem is deterministic). By recovering the constant in the proof, we can show that

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \frac{1}{N}\left(\frac{\alpha\Omega^2}{\eta} + \sum_{n=1}^{N}\left(\frac{1}{2}\frac{L^2\eta}{\alpha} - \frac{\alpha}{2\eta}\right)\|x_n - \hat{x}_n\|^2\right)$$

Suppose . We choose η to make the second term non-positive, i.e.

$$\frac{1}{2}\frac{L^2\eta}{\alpha} - \frac{\alpha}{2\eta} \leq 0 \iff \eta \leq \frac{\alpha}{L}$$

and the error bound becomes

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \frac{L\Omega^2}{N}$$

This bound and the condition on η matches that in (Ho-Nguyen and Kılınç-Karzan, 2018).

Stochastic Problems For stochastic problems, we use the condition specified in Theorem 7.E.4. Suppose $2p - k > -1$. To balance the second term in (7.19) and (7.20), we choose

$$2p - k + 1 = k \implies k = p + \frac{1}{2}$$

To satisfy the hypothesis $2p-k > -1$, it requires $p > -\frac{1}{2}$. Note with this choice, it satisfies the condition $k > p$ required in Theorem 7.E.4. Therefore, the overall bound becomes

$$\begin{aligned}
\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] &\leq \frac{1}{w_{1:N}} \left(\frac{\alpha\Omega^2}{\eta} N^{p+\frac{1}{2}} + \frac{3\sigma^2\eta}{\alpha} \sum_{n=1}^{N} n^{p-\frac{1}{2}} \right) + \frac{O(1)}{w_{1:N}} \\
&\leq \frac{p+1}{N^{p+1}} \left(\frac{\alpha\Omega^2}{\eta} + \frac{3\eta\sigma^2}{\alpha(p+\frac{1}{2})} \right) (N+1)^{p+\frac{1}{2}} + \frac{O(1)}{N^{p+1}} \\
&\leq e^{\frac{p+1/2}{N}} (p+1) \left(\frac{\alpha\Omega^2}{\eta} + \frac{3\eta\sigma^2}{\alpha(p+\frac{1}{2})} \right) N^{-\frac{1}{2}} + \frac{O(1)}{N^{p+1}}
\end{aligned}$$

where we use Lemma 7.G.1 and $(\frac{N+1}{N})^{p+1/2} \leq e^{\frac{p+1/2}{N}}$. If we set η such that

$$\frac{\alpha\Omega^2}{\eta} = \frac{3\eta\sigma^2}{\alpha(p+\frac{1}{2})} \implies \eta = \alpha \frac{\Omega}{\sigma} \sqrt{\frac{p+\frac{1}{2}}{3}}$$

Then

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq 2e^{\frac{p+1/2}{N}} (p+1)\Omega\sigma \sqrt{\frac{3}{p+\frac{1}{2}}} N^{-\frac{1}{2}} + \frac{O(1)}{N^{p+1}} \tag{7.21}$$

For example, if $p = 0$, then

$$\mathbb{E}[\mathrm{ERR}(\bar{x}_N)] \leq \frac{O(1)}{N} + \frac{2\sqrt{6}\sigma\Omega e^{\frac{p+1/2}{N}}}{\sqrt{N}}$$

which matches the bound in by Juditsky, Nemirovski, and Tauvel (2011) with a slightly worse constant. We leave a complete study of tuning p as future work.

7.E.4 Comparison of stochastic MIRROR-PROX and MOBIL-PROX in Imitation Learning

The major difference between stochastic MIRROR-PROX and MOBIL-PROX is whether the gradient from the environment is used to also update the decision π_{n+1}. It is used in the MIRROR-PROX, whereas MOBIL-PROX uses the estimation from simulation. Therefore,

for N iterations, MOBIL-PROX requires only N interactions, whereas MIRROR-PROX requires $2N$ interactions.

The price MOBIL-PROX pays extra when using the estimated gradient is that a secondary online learning problem has to be solved. This shows up in the term, for example of strongly convex problems,

$$\frac{(p+1)G_h^2}{2\mu_h}\frac{1}{N^2} + \frac{\epsilon_{\hat{\mathcal{F}}}^w + \sigma_g^2 + \sigma_{\hat{g}}^2}{N}$$

in Theorem 7.4.2. If both gradients are from the environment, then $\epsilon_{\hat{\mathcal{F}}}^w = 0$ and $\sigma_{\hat{g}}^2 = \sigma_g^2$. Therefore, if we ignore the $O(\frac{1}{N^2})$ term, using an estimated gradient to update π_{n+1} is preferred, if it requires less interactions to get to the magnitude of error, i.e.

$$2 \times 2\sigma_g^2 \geq \epsilon_{\hat{\mathcal{F}}}^w + \sigma_{\hat{g}}^2 + \sigma_g^2$$

in which the multiplier of 2 on the left-hand side is due to MOBIL-PROX only requires one interaction per iterations, whereas stochastic MIRROR-PROX requires two.

Because σ_g^2 is usually large in real-world RL problems and $\sigma_{\hat{g}}^2$ can be made close to zero easily (by running more simulations), if our model class is reasonably expressive, then MOBIL-PROX is preferable. Essentially, this is because MOBIL-PROX can roughly cut the noise of gradient estimates by half.

The preference over MOBIL-PROX would be more significant for convex problems, because the error decays slower over iterations (e.g. $\frac{1}{\sqrt{N}}$) and therefore more iterations are required by the stochastic MIRROR-PROX approach to counter balance the slowness due to using noisy gradient estimator.

7.F Experimental Details

7.F.1 Tasks

Two robot control tasks (Cartpole and Reacher3D) powered by the DART physics engine [19] were used as the task environments.

Cartpole The Cart-Pole Balancing task is a classic control problem, of which the goal is to keep the pole balanced in an upright posture with force only applied to the cart. The state and action spaces are both continuous, with dimension 4 and 1, respectively. The state includes the horizontal position and velocity of the cart, and the angle and angular velocity of the pole. The time-horizon of this task is 1000 steps. There is a small uniformly random perturbation injected to initial state, and the transition is deterministic. The agent receives $+1$ reward for every time step it stays in a predefined region, and a rollout terminates when the agent steps outside the region.

Reacher3D In this task, a 5-DOF (degrees-of-freedom) manipulator is controlled to reach a random target position in a 3D space. The reward is the sum of the negative distance to the target point from the finger tip and a control magnitude penalty. The actions correspond to the torques applied to the 5 joints. The time-horizon of this task is 500 steps. At the beginning of each rollout, the target point to reach is reset to a random location.

7.F.2 Algorithms

Policies We employed Gaussian policies in our experiments, i.e. for any state $s \in \mathbb{S}$, π_s is Gaussian distributed. The mean of π_s was modeled by either a linear function or a neural network that has 2 hidden layers of size 32 and $\tanh$ activation functions. The covariance matrix of π_s was restricted to be diagonal and independent of state. The expert policies in the IL experiments share the same architecture as the corresponding learners (e.g. a

linear learner is paired with a linear expert) and were trained using actor-critic-based policy gradients.

Imitation learning loss With regard to the IL loss, we set $D(\pi_s^\star||\pi_s)$ in (7.2) to be the KL-divergence between the two Gaussian distributions: $D(\pi_s^\star||\pi_s) = KL[\pi_s||\pi_s^\star]$. (We observed that using $KL[\pi_s||\pi_s^\star]$ converges noticeably faster than using $KL[\pi_s^\star||\pi_s]$).

Implementation details of MOBIL-PROX The regularization of MOBIL-PROX was set to $r_n(\pi) = \frac{\mu_l \alpha_n}{2}\|\pi - \pi_n\|^2$ such that $\sum w_n \alpha_n \mu_l = (1 + cn^{p+1/2})/\eta_n$, where $c = 0.1$ and η_n was adaptive to the norm of the prediction error. Specifically, we used $\eta_n = \eta\lambda_n$: $\eta > 0$ and λ_n is a moving-average estimator of the norm of $e_n = g_n - \hat{g}_n$ defined as

$$\bar{\lambda}_n = \beta\bar{\lambda}_{n-1} + (1-\beta)\|e_n\|_2$$
$$\lambda_n = \bar{\lambda}_n/(1-\beta^n)$$

where β was chosen to be 0.999. This parameterization is motivated by the form of the optimal step size of MOBIL-PROX in Theorem 7.4.2, and by the need of having adaptive step sizes so different algorithms are more comparable. The model-free setting was implemented by setting $\hat{g}_n = 0$ in MOBIL-PROX, and the same adaptation rule above was used (which in this case effectively adjusts the learning rate based on $\|g_n\|$). In the experiments, η was selected to be 0.1 and 0.01 for $p = 0$ and $p = 2$, respectively, so the areas under the effective learning rate $\eta_n w^p/(1 + cn^{p+1/2})$ for $p = 0$ and $p = 2$ are close, making MOBIL-PROX perform similarly in these two settings.

In addition to the update rule of MOBIL-PROX, a running normalizer, which estimates the upper and the lower bounds of the state space, was used to center the state before it was fed to the policies.

Dynamics model learning The dynamics model used in the experiments is deterministic (the true model is deterministic too). It is represented by a neural network with 2 hidden layers of size 64 and $\tanh$ activation functions. Given a batch of transition triples $\{(s_{t_k}, a_{t_k}, s_{t_k+1})\}_{k=1}^{K}$ collected by running π_n under the true dynamics in each round, we set the per-round loss for model learning as $\frac{1}{K}\sum_{k=1}^{K} \|s_{t_{k+1}} - M(s_{t_k}, a_{t_k})\|_2^2$, where M is the neural network dynamics model. It can be shown that this loss is an upper bound of $\|\nabla_2 F(\pi_n, \pi_n) - \nabla_2 \hat{F}_n(\pi_n, \pi_n)\|_*^2$ by applying a similar proof as in Section 7.C. The minimization was achieved through gradient descent using ADAM Kingma and Ba, 2014 with a fixed number of iterations (2048) and fixed-sized mini-batches (128). The step size of ADAM was set to 0.001.

7.G Useful Lemmas

This section summarizes some useful properties of polynomial partial sum, sequence in Banach space, and variants of FTL in online learning. These results will be useful to the proofs in Section 7.B.

7.G.1 Polynomial Partial Sum

Lemma 7.G.1. *This lemma provides estimates of* $\sum_{n=1}^{N} n^p$.

1. *For* $p > 0$, $\frac{N^{p+1}}{p+1} = \int_0^N x^p dx \leq \sum_{n=1}^{N} n^p \leq \int_1^{N+1} x^p dx \leq \frac{(N+1)^{p+1}}{p+1}$.
2. *For* $p = 0$, $\sum_{n=1}^{N} n^p = N$.
3. *For* $-1 < p < 0$,

$$\frac{(N+1)^{p+1}-1}{p+1} = \int_1^{N+1} x^p dx \leq \sum_{n=1}^{N} n^p \leq 1 + \int_1^N x^p dx = \frac{N^{p+1}+p}{p+1} \leq \frac{(N+1)^{p+1}}{p+1}.$$

4. *For* $p = -1$, $\ln(N+1) \leq \sum_{n=1}^{N} n^p \leq \ln N + 1$.
5. *For* $p < -1$, $\sum_{n=1}^{N} n^p \leq \frac{N^{p+1}+p}{p+1} = O(1)$. *For* $p = -2$, $\sum_{n=1}^{N} n^p \leq \frac{N^{-1}-2}{-2+1} \leq 2$.

Lemma 7.G.2. *For* $p \geq -1$, $N \in \mathbb{N}$,

$$S(p) = \sum_{n=1}^{N} \frac{n^{2p}}{\sum_{m=1}^{n} m^p} \leq \begin{cases} \frac{p+1}{p}(N+1)^p, & \textit{for } p > 0 \\ \ln(N+1), & \textit{for } p = 0 \\ O(1), & \textit{for } -1 < p < 0 \\ 2, & \textit{for } p = -1 \end{cases}.$$

Proof. If $p \geq 0$, by Lemma 7.G.1,

$$S(p) = (p+1)\sum_{n=1}^{N} n^{p-1} \leq \begin{cases} \frac{p+1}{p}(N+1)^p, & \text{for } p > 0 \\ \ln(N+1), & \text{for } p = 0 \end{cases}.$$

If $-1 < p < 0$, by Lemma 7.G.1, $S(p) \leq (p+1)\sum_{n=1}^{N} \frac{n^{2p}}{(n+1)^{p+1}-1}$. Let $a_n = \frac{n^{2p}}{(n+1)^{p+1}-1}$, and $b_n = n^{p-1}$. Since $\lim_{n\to\infty} \frac{a_n}{b_n} = 1$ and by Lemma 7.G.1 $\sum_{n=0}^{\infty} b_n$ converges, thus $\sum_{n=0}^{\infty} a_n$ converges too. Finally, if $p = -1$, by Lemma 7.G.1, $S(-1) \leq \sum_{n=1}^{N} \frac{1}{n^2 \ln(n+1)} \leq \sum_{n=1}^{N} \frac{1}{n^2} \leq 2$. ■

7.G.2 Sequence in Banach Space

Lemma 7.G.3. *Let* $\{a = x_0, x_1, \cdots, x_N = b\}$ *be a sequence in a Banach space with norm* $\|\cdot\|$. *Then for any* $N \in \mathbb{N}_+$, $\|a-b\|^2 \leq N\sum_{n=1}^{N} \|x_{n-1} - x_n\|^2$.

Proof. First we note that by triangular inequality it satisfies that $\|a-b\| \leq \sum_{n=1}^{N} \|x_{n-1} - x_n\|$. Then we use the basic fact that $2ab \leq a^2 + b^2$ in the second inequality below and prove the result.

$$\begin{aligned}
\|a-b\|^2 &\leq \sum_{n=1}^{N} \|x_{n-1} - x_n\|^2 + \sum_{n=1}^{N} \sum_{m=1;m\neq n}^{N} \|x_{n-1} - x_n\| \|x_{m-1} - x_m\| \\
&\leq \sum_{n=1}^{N} \|x_{n-1} - x_n\|^2 + \sum_{n=1}^{N} \sum_{m=1;m\neq n}^{N} \frac{1}{2}\left(\|x_{n-1} - x_n\|^2 + \|x_{m-1} - x_m\|^2\right) \\
&= \sum_{n=1}^{N} \|x_{n-1} - x_n\|^2 + \frac{N-1}{2}\sum_{n=1}^{N} \|x_{n-1} - x_n\|^2 + \frac{1}{2}\sum_{n=1}^{N} \sum_{m=1;m\neq n}^{N} \|x_{m-1} - x_m\|^2 \\
&= \sum_{n=1}^{N} \|x_{n-1} - x_n\|^2 + (N-1)\sum_{n=1}^{N} \|x_{n-1} - x_n\|^2 \\
&= N\sum_{n=1}^{N} \|x_{n-1} - x_n\|^2
\end{aligned}$$

■

7.G.3 Basic Regret Bounds of Online Learning

For this chapter to be self-contained, we summarize some fundamental results of regret bound when the learner in an online problem updates the decisions by variants of FTL. Here we consider a general setup and therefore use a slightly different notation from the one used in the main part of this chapter for policy optimization.

Online Learning Setup Consider an online convex optimization problem. Let $\mathcal{X}$ be a compact decision set in a normed space with norm $\|\cdot\|$. In round n, the learner plays $x_n \in \mathcal{X}$ receives a convex loss $l_n : \mathcal{X} \to \mathbb{R}$ satisfying $\|\nabla l_n(x_n)\|_* \leq G$, and then make a new decision $x_{n+1} \in \mathcal{X}$. The *regret* is defined as

$$\mathrm{regret}_N(\mathcal{X}) = \sum_{n=1}^{N} l_n(x_n) - \min_{x \in \mathcal{X}} \sum_{n=1}^{N} l_n(x)$$

More generally, let $\{w_n \in \mathbb{R}_+\}_{n=1}^N$ be a sequence of weights. The *weighted regret* is defined as

$$\mathrm{regret}_N^w(\mathcal{X}) = \sum_{n=1}^{N} w_n l_n(x_n) - \min_{x \in \mathcal{X}} \sum_{n=1}^{N} w_n l_n(x)$$

In addition, we define a constant $\epsilon_{\mathcal{X}}^w$ (which can depend on $\{l_n\}_{n=1}^N$) such that

$$\epsilon_{\mathcal{X}}^w \geq \min_{x \in \mathcal{X}} \frac{\sum_{n=1}^N w_n l_n(x)}{w_{1:N}}.$$

In the following, we prove some basic properties of FTL with prediction. At the end, we show the result of FTL as a special case. These results are based on the Strong FTL Lemma (Lemma 10.F.2), which can also be proven by Stronger FTL Lemma (Lemma 10.F.3).

Lemma 10.F.2 (Strong FTL Lemma (McMahan, 2017))**.** *For any sequence of decisions* $\{x_n \in \mathcal{X}\}$ *and loss functions* $\{\zeta_n\}$, $\mathrm{regret}_N(\mathcal{X}) \leq \sum_{n=1}^N \zeta_{1:n}(x_n) - \zeta_{1:n}(x_n^\star)$, *where* $x_n^\star \in \arg\min_{x \in \mathcal{X}} \zeta_{1:n}(x)$, *where* $\mathcal{X}$ *is the decision set.*

To use Lemma 10.F.2, we first show an intermediate bound.

Lemma 7.G.4. *In round n, let $l_{1:n}$ be $\mu_{1:n}$-strongly convex for some $\mu_{1:n} > 0$, and let v_{n+1} be a (non)convex function such that $l_{1:n} + v_{n+1}$ is convex. Suppose the learner plays FTL with prediction, i.e. $x_{n+1} \in \arg\min_{x \in \mathcal{X}} \left(l_{1:n} + v_{n+1}\right)(x)$. Then it holds*

$$\sum_{n=1}^{N} \left(l_{1:n}(x_n) - l_{1:n}(x_n^\star)\right) \leq \sum_{n=1}^{N} \frac{1}{2\mu_{1:n}} \|\nabla l_n(x_n) - \nabla v_n(x_n)\|_*^2$$

where $x_n^\star = \arg\min_{\mathcal{X}} \sum_{n=1}^{N} l_n(x)$.

Proof. For any $x \in \mathcal{X}$, since $l_{1:n}$ is $\mu_{1:n}$ strongly convex, we have

$$l_{1:n}(x_n) - l_{1:n}(x) \leq \langle \nabla l_{1:n}(x_n), x_n - x \rangle - \frac{\mu_{1:n}}{2} \|x_n - x\|^2. \tag{7.22}$$

And by the hypothesis $x_n = \arg\min_{x \in \mathcal{X}} \left(l_{1:n-1} + v_n\right)(x)$, it holds that

$$\langle -\nabla l_{1:n-1}(x_n) - \nabla v_n(x_n), x_n - x \rangle \geq 0. \tag{7.23}$$

Adding (7.22) and (7.23) yields

$$\begin{aligned} l_{1:n}(x_n) - l_{1:n}(x) &\leq \langle \nabla l_n(x_n) - \nabla v_n(x_n), x_n - x \rangle - \frac{\mu_{1:n}}{2} \|x_n - x\|^2 \\ &\leq \max_d \langle \nabla l_n(x_n) - \nabla v_n(x_n), d \rangle - \frac{\mu_{1:n}}{2} \|d\|^2 \\ &= \frac{1}{2\mu_{1:n}} \|\nabla l_n(x_n) - \nabla v_n(x_n)\|_*^2, \end{aligned}$$

where the last equality is due to a property of dual norm (e.g. Exercise 3.27 of Boyd and Vandenberghe, 2004). Substituting $x_n^\star$ for x and taking the summation over n prove the lemma. ■

Using Lemma 10.F.2 and Lemma 7.G.4, we can prove the regret bound of FTL with prediction.

Lemma 7.G.5 (FTL with prediction)**.** *Let l_n be a μ_n-strongly convex for some $\mu_n \geq 0$. In round n, let v_{n+1} be a (non)convex function such that $\sum_{m=1}^{n} w_m l_m + w_{m+1} v_{n+1}$ is convex. Suppose the learner plays FTL with prediction, i.e. $x_{n+1} \in \arg\min_{x \in \mathcal{X}} \sum_{m=1}^{n} (w_m l_m + w_{m+1} v_{n+1})(x)$ and suppose that $\sum_{m=1}^{n} w_m \mu_m > 0$. Then*

$$\operatorname{regret}_N^w(\mathcal{X}) \leq \sum_{n=1}^{N} \frac{w_n^2 \|\nabla l_n(x_n) - \nabla v_n(x_n)\|_*^2}{2 \sum_{m=1}^{n} w_m \mu_m}$$

In particular, if $\mu_n = \mu$, $w_n = n^p$, $p \geq 0$, $\operatorname{regret}_N^w(\mathcal{X}) \leq \frac{p+1}{2\mu} \sum_{n=1}^{N} n^{p-1} \|\nabla l_n(x_n) - \nabla v_n(x_n)\|_*^2$.

Proof. By Lemma 10.F.2 and Lemma 7.G.4, we see

$$\operatorname{regret}_N^w(\mathcal{X}) \leq \sum_{n=1}^{N} (l_{1:n}(x_n) - l_{1:n}(x_n^\star)) \leq \sum_{n=1}^{N} \frac{w_n^2 \|\nabla l_n(x_n) - \nabla v_n(x_n)\|_*^2}{2 \sum_{m=1}^{n} w_m \mu_m}.$$

If $\mu_n = \mu$, $w_n = n^p$, and $p \geq 0$, then it follows from Lemma 7.G.1

$$\operatorname{regret}_N^w(\mathcal{X}) \leq \frac{1}{2\mu} \sum_{n=1}^{N} \frac{n^{2p}}{\frac{n^{p+1}}{p+1}} \|\nabla l_n(x_n) - \nabla v_n(x_n)\|_*^2 = \frac{p+1}{2\mu} \sum_{n=1}^{N} n^{p-1} \|\nabla l_n(x_n) - \nabla v_n(x_n)\|_*^2.$$

■

The next lemma about the regret of FTL is a corollary of Lemma 7.G.5.

Lemma 7.G.6 (FTL)**.** *Let l_n be μ-strongly convex for some $\mu > 0$. Suppose the learner play FTL, i.e. $x_n = \arg\min_{x \in \mathcal{X}} \sum_{m=1}^{n} w_m l_m(x)$. Then $\operatorname{regret}_N^w(\mathcal{X}) \leq \frac{G^2}{2\mu} \sum_{n=1}^{N} \frac{w_n^2}{w_{1:n}}$. In*

particular, if $w_n = n^p$*, then*

$$\sum_{n=1}^{N} w_n l_n(x_n) \leq \begin{cases} \frac{G^2}{2\mu}\frac{p+1}{p}(N+1)^p + \frac{1}{p+1}(N+1)^{p+1}\epsilon_{\mathcal{X}}^w, & \textit{for } p > 0 \\ \frac{G^2}{2\mu}\ln(N+1) + N\epsilon_{\mathcal{X}}^w, & \textit{for } p = 0 \\ \frac{G^2}{2\mu}O(1) + \frac{1}{p+1}(N+1)^{p+1}\epsilon_{\mathcal{X}}^w, & \textit{for } -1 < p < 0 \\ \frac{G^2}{\mu} + (\ln N + 1)\epsilon_{\mathcal{X}}^w, & \textit{for } p = -1 \end{cases}$$

Proof. By definition of $\mathrm{regret}_N^w(\mathcal{X})$, the absolute cost satisfies $\sum_{n=1}^N w_n l_n(x_n) = \mathrm{regret}_N^w(\mathcal{X}) + \min_{x\in\mathcal{X}} \sum_{n=1}^N w_n l_n(x)$. We bound the two terms separately. For $\mathrm{regret}_N^w(\mathcal{X})$, set $v_n = 0$ in Lemma 7.G.5 and we have

$$\begin{aligned} \mathrm{regret}_N^w(\mathcal{X}) &\leq \frac{G^2}{2\mu}\sum_{n=1}^N \frac{w_n^2}{w_{1:n}} && \text{(Lemma 7.G.5 and gradient bound)} \\ &= \frac{G^2}{2\mu}\sum_{n=1}^N \frac{n^{2p}}{\sum_{m=1}^n m^p} && \text{(Special case } w_n = n^p\text{)}, \end{aligned}$$

in which $\sum_{n=1}^N \frac{n^{2p}}{\sum_{m=1}^n m^p}$ is exactly what 7.G.2 bounds. On the other hand, the definition of $\epsilon_{\mathcal{X}}^w$ implies that $\min_{x\in\mathcal{X}} \sum_{n=1}^N w_n l_n(x) \leq w_{1:N}\epsilon_{\mathcal{X}}^w = \sum_{n=1}^N n^p \epsilon_{\mathcal{X}}^w$, where $\sum_{n=1}^N n^p$ is bounded by Lemma 7.G.1. Combining these two bounds, we conclude the lemma. ■

Part II

Policy Optimization II: Abstraction

ABSTRACT

In Part I of the thesis, we improved the state-of-the-art results of imitation learning (IL). This advancement is made by the online-learning-style analyses and the observation that these practical online learning problems possess certain regularity that has been ignored in the standard abstract online learning analyses. Therefore, a natural follow-up question to this research is whether such ideas can be generalized to problems outside IL (Cheng et al., 2019a,c,d).

In Chapter 8, we establish a formal, abstract setup of this class of problems, which we name Continuous Online Learning (COL). We show that COL covers and more appropriately describes many interesting applications, from general equilibrium problems (EPs) to optimization in episodic MDPs, including the online IL discussed in Chapter 6. To demonstrate the potential of COL, we revisit the classic linear program setups of RL in Chapter 9, and provide a reduction from RL to COL, by which *any* online algorithm with sublinear regret can generate policies with provable performance guarantees.

Complementary to the refined COL setup is the meta algorithm PICCOLO presented in Chapter 10. Based on the concept of predictive models introduced in Chapter 7, PICCOLO provides a constructive way to design algorithms that leverage the predictability in losses to accelerate online learning. By casting policy optimization as online learning, we can use PICCOLO as a hybrid algorithm to combines model-based and model-free updates, where the concept of predictive model serves an abstraction of model information (i.e. past experiences). Importantly, we show that PICCOLO does not suffer from policy performance bias due to modeling error, unlike the classic model-based approaches. Therefore, PICCOLO provides a framework for designing new algorithms that can safely leverage imperfect models.

CHAPTER 8

ONLINE LEARNING WITH CONTINUOUS VARIATIONS

8.1 Introduction

Online learning (Gordon, 1999; Zinkevich, 2003), which studies the interactions between a learner (i.e. an algorithm) and an opponent through regret minimization, has proved to be a powerful framework for analyzing and designing iterative algorithms. However, while classic setups focus on bounding the worst case, many applications are not naturally adversarial. In this work, we aim to bridge this reality gap by establishing a new online learning setup that better captures certain regularity that appears in practical problems.

Formally, we recall an online learning problem repeats the following steps: in round n, the learner plays a decision x_n from a decision set $\mathcal{X}$, the opponent chooses a loss function $l_n : \mathcal{X} \to \mathbb{R}$ based on the decisions of the learner, and then information about l_n (e.g. $\nabla l_n(x_n)$) is revealed to the learner for making the next decision. This abstract setup (Hazan, 2016; Shalev-Shwartz, 2012) studies the *adversarial* setting where l_n can be almost arbitrarily chosen except for minor restrictions like convexity. Often the performance is measured relatively through *static regret*,

$$\text{regret}_N^s \coloneqq \sum_{n=1}^N l_n(x_n) - \min_{x \in \mathcal{X}} \sum_{n=1}^N l_n(x). \tag{8.1}$$

Recently, interest has emerged in algorithms that make decisions that are nearly optimal at each round. The regret is therefore measured on-the-fly and suitably named *dynamic regret*,

$$\text{regret}_N^d \coloneqq \sum_{n=1}^N l_n(x_n) - \sum_{n=1}^N l_n(x_n^*), \tag{8.2}$$

where $x_n^* \in \arg\min_{x \in \mathcal{X}} l_n(x)$. As dynamic regret by definition upper bounds static regret, minimizing the dynamic regret is a more difficult problem.

While algorithms with sublinear static regret are well understood, the research on dynamic regret is relatively recent. As dynamic regret grows linearly in the adversarial setup, most papers (Besbes, Gur, and Zeevi, 2015; Dixit et al., 2019; Jadbabaie et al., 2015; Mokhtari et al., 2016; Yang et al., 2016; Zhang et al., 2017; Zinkevich, 2003) focus on how dynamic regret depends on certain variations of the loss sequence across rounds (such as the path variation $V_N = \sum_{n=1}^{N-1} \|x_n^* - x_{n+1}^*\|$). Even if the algorithm does not require knowing the variation, the bound is still written in terms of it. While tight bounds have been established (Yang et al., 2016), their results do not always translate into conditions for achieving sublinear dynamic regret in practice, because the size (budget) of the variations can be difficult to verify beforehand. This is especially the case when the opponent is *adaptive*, responding the learner's decisions at each round. In these situations, it is unknown if existing results become vacuous or yield sublinear dynamic regret.

Motivated by the use of online learning to analyze iterative algorithms in practice, we consider a new setup we call Continuous Online Learning (COL), which directly models regularity in losses as part of the problem definition, as opposed to the classic adversarial setup that adds ad-hoc budgets. As we will see, this minor modification changes how regret and feedback interact and makes the quest of seeking sublinear dynamic regret well defined and interpretable even for adaptive opponents, without imposing variation budgets.

8.1.1 Definition of COL

We describe COL as follows. We suppose that the opponent possesses a bifunction $f : (x, x') \mapsto f_x(x') \in \mathbb{R}$, for $x, x' \in \mathcal{X}$, that is *unknown* to the learner. This bifunction is used by the opponent to determine the per-round losses: in round n, if the learner chooses x_n,

then the opponent responds with

$$l_n(x) = f_{x_n}(x). \tag{8.3}$$

Finally, the learner suffers $l_n(x_n)$ and receives feedback about l_n. For $f_x(x')$, we treat x as the *query argument* that proposes a question (i.e. an optimization objective $f_x(\cdot)$), and treat x' as the *decision argument* whose performance is evaluated. This bifunction f generally can be defined online as queried, with only one limitation that the same loss function $f_x(\cdot)$ must be selected by the opponent whenever the learner plays the same decision x. Thus, the opponent can be adaptive, but in response to only the learner's current decision.

In addition to the restriction in (8.3), we impose regularity into f to relate l_n across rounds, so that seeking sublinear dynamic regret becomes well defined.[1]

Definition 8.1.1. We say an online learning problem is *continuous* if l_n is set as in (8.3) by a bifunction f satisfying, $\forall x' \in \mathcal{X}$, $\nabla f_x(x')$ is a continuous map in x [2].

The continuity structure in Definition 8.1.1 and the constraint (8.3) in COL limit the degree that losses can vary, making it possible for the learner to partially infer future losses from the past experiences.

The continuity may appear to restrict COL to purely deterministic settings, but adversity such as stochasticity can be incorporated via an important *nuance* in the relationship between loss and feedback. In the classical online learning setting, the adversity is incorporated in the loss: the losses l_n and decisions x_n may themselves be generated adversarially or stochastically and then they directly determine the feedback, e.g., given as full information (receiving l_n or $\nabla l_n(x_n)$) or bandit (just $l_n(x_n)$). The (expected) regret is then measured with respect to these intrinsically adversarial losses l_n. By contrast, in COL, we always measure regret with respect to the true underlying bifunction $l_n = f_{x_n}$. However, we give the opponent the freedom to add an additional stochastic or adversarial component into

[1] Otherwise the opponent can define $f_x(\cdot)$ pointwise for each x to make $l_n(x_n) - l_n(x_n^*)$ constant.

[2] We define $\nabla f_x(x')$ as the derivative with respect to x'.

the feedback; e.g., in first-order feedback, the learner could receive $g_n = \nabla l_n(x_n) + \xi_n$, where ξ_n is a probabilistically bounded and potentially adversarial vector, which can be used to model noise or bias in feedback. In other words, the COL setting models a true underlying loss with regularity, but allows adversary to be modeled within the feedback. This addition is especially important for dynamic regret, as it allows us to always consider regret against the true f_{x_n} while still incorporating the possibility of stochasticity.

8.1.2 Examples

At this point, the setup of COL may sound restrictive, but this setting is in fact motivated by a general class of problems and iterative algorithms used in practice, some of which have been previously analyzed in the online learning setting. Generally, COL describes the trial-and-error principle, which attempts to achieve a difficult objective $f_x(x)$ through iteratively constructing a sequence of simplified and related subproblems $f_{x_n}(x)$, similar to majorize-minimize (MM) algorithms. Our first application of this kind is the use of iterative algorithms in solving (stochastic) equilibrium problems (EPs) (Bianchi and Schaible, 1996). EP is a well-studied subject in mathematical programming, which includes optimization, saddle-point problems, variational inequality (VI) (Facchinei and Pang, 2007), fixed-point problem (FP), etc. Except for toy cases, these problems usually rely on using iterative algorithms to generate ϵ-approximate solutions; interestingly these algorithms often resemble known algorithms in online learning, such as mirror descent or Follow-the-Leader (FTL). In Sections 8.4 and 8.5, we will show that how the residual function of these problems renders a natural choice of bifunction f in COL, and how the regret of COL relates to its solution quality. In this example, it is particularly important to classify the adversary (e.g. due to bias or stochasticity) as feedback rather than loss function, in order to properly incorporate the continuity in the source problem.

Another class of interesting COL problems comes from optimization in episodic Markov decision processes (MDPs). In online imitation learning (Ross, Gordon, and Bagnell, 2011)

that we discussed in Part I of the thesis, the learner optimizes a policy to mimic an expert $\pi^\star$. In round n, the loss is $l_n(\pi) = \mathbb{E}_{s\sim d^{\pi_n}}[c(s, \pi; \pi^\star)]$, where d^{π_n} is the state distribution visited by running the learner's policy π_n in the MDP, and $c(s, \pi; \pi^\star)$ is a cost that measures the difference between a policy π and the expert $\pi^\star$. This is a bifunction form where continuity exists due to expectation and feedback is noisy about l_n (allowed by our feedback model). In fact, online IL is the main inspiration behind this research. An early analysis of IL was framed using the adversarial, static regret setup (Ross, Gordon, and Bagnell, 2011). Recently, results were refined through the use of continuity in the bifunction and dynamic regret (Cheng and Boots, 2018; Cheng et al., 2019b; Lee et al., 2018c). This problem again highlights the importance of treating stochasticity as the feedback. We wish to measure regret with respect to the expected cost $l_n(\pi)$ which admits a continuous structure, but feedback only arrives via stochastic samples from the MDP. Structural prediction and system identification can be framed similarly (Ross and Bagnell, 2012; Venkatraman, Hebert, and Bagnell, 2015).

Lastly, we note that the classic fitted Q-iteration (Gordon, 1995; Riedmiller, 2005) for reinforcement learning also uses a similar setup. In the n round, the loss can be written as $l_n(Q) = \mathbb{E}_{s,a\sim\mu^{\pi(Q_n)}}\mathbb{E}_{s'\sim\mathcal{P}|s,a}[(Q(s, a) - r(s, a) - \gamma \max_{a'} Q_n(s', a'))^2]$, where $\mu^{\pi(Q_n)}$ is the state-action distribution[3] induced by running a policy $\pi(Q_n)$ based on the Q-function Q_n of the learner, and $\mathcal{P}$ is the transition dynamics, r is the reward, and γ is the discount factor. Again this is a COL problem.

8.1.3 Main Results

The goal of this chapter is to establish COL and to study, particularly, conditions and efficient algorithms for achieving sublinear dynamic regret. We choose not to pursue algorithms with fast static regret rates in COL, as there have been studies on how algorithms can systematically leverage continuity in COL to accelerate learning (Cheng et al., 2019b,d)

[3] Or some fixed distribution with sufficient excitation.

though they are framed as online IL research. On the contrary, the knowledge about dynamic regret is less known, except for (Cheng and Boots, 2018; Lee et al., 2018c) (also framed as online IL) which study the convergence of FTL and mirror descent, respectively.

Our first result shows that achieving sublinear dynamic regret in COL, interestingly, is equivalent to solving certain EP, VI, and FP problems, which are known to be PPAD-complete[4] (Daskalakis, Goldberg, and Papadimitriou, 2009). In other words, generally, achieving sublinear dynamic regret that is polynomial in the dimension of the decision set can be extremely difficult.

Nevertheless, based on the solution concept of EP, VI, and FP, we show a reduction from monotone EPs to COL, and present necessary conditions and sufficient conditions for achieving sublinear dynamic regrets with polynomial dependency. Particularly, we show a *reduction* from sublinear dynamic regret to static regret and convergence to the solution of the EP/VI/FP. This reduction allows us to quickly derive non-asymptotic dynamic regret bounds of popular online learning algorithms based on their known static regret rates. At the end, we extend COL to consider partially adversarial loss and discuss open questions.

8.2 Related Work

Much work in the dynamic regret has focused on improving rates with respect to various measures of the loss sequence's variation. Mokhtari et al., 2016; Zinkevich, 2003 showed that the dynamic regret of gradient descent in terms of the path variation. Other measures of variation such as functional variation (Besbes, Gur, and Zeevi, 2015) and squared path variation (Zhang et al., 2017) have also been studied. While these algorithms may not need to know the variation size beforehand, their guarantees are still stated in terms of these variations. Therefore, these results can be difficult to interpret, when the losses can be chosen adaptively.

[4] In short, they are NP problems whose solutions are known to exist, but it is open as to if they belong to P.

To illustrate, consider the online IL problem. It is impossible to know the variation budget *a priori* because the loss observed at each round is a function of the policy selected by the algorithm. This budget could easily be linear, if an algorithm selects very disparate policies, or it could be zero if the algorithm always naively returns the same policy. Thus, existing budget-based results cannot tell the convergence of an IL algorithm.

Our work is also closely related to that of (Hall and Willett, 2013; Rakhlin and Sridharan, 2012), which consider *predictable* loss sequences, i.e. sequences that are presumed to be non-adversarial and admit improved regret rates. The former considers static regret for both full and partial information cases, and the latter considers a similar problem setting but for the dynamic regret case. These analyses, however, still require a known variation quantity in order to be interpretable.

By contrast, we leverage extra structures of COL to provide interpretable dynamic regret rates, without *a priori* constraints on the variation. That is, our rates are internally governed by the algorithms, rather than externally dictated by a variation budget. This problem setup in some sense is more difficult as achieving sublinear dynamic regret here requires both the per-round losses and the loss variation, as a function of the learner's decisions, are *simultaneously* small. Nonetheless, we can show conditions for sublinear dynamic regret, using the bifunction structure in COL.

8.3 Preliminaries

We review background, in particular VIs and EPs, for completeness (Bianchi and Schaible, 1996; Facchinei and Pang, 2007; Konnov and Laitinen, 2002).

Notation Throughout the paper, we reserve the notation f to denote the bifunction that defines COL problems, and we assume $\mathcal{X} \subset \mathbb{R}^d$ is compact and convex, where $d \in \mathbb{N}_+$ is finite. We equip $\mathcal{X}$ with norm $\|\cdot\|$, which is not necessarily Euclidean, and write $\|\cdot\|_*$ to denote its dual norm. We denote its diameter by $D_{\mathcal{X}} \coloneqq \max_{x,x' \in \mathcal{X}} \|x - x'\|$.

As in the usual online learning, we are particularly interested in the case where $f_x(\cdot)$ is convex and continuous. For simplicity, we will assume all functions are continuously differentiable, except for $f_x(x')$ as a function over the querying argument x, where $x' \in \mathcal{X}$. We will use ∇ to denote gradients. In particular, for the bifunction f, we use ∇f to denote $\nabla f : x \mapsto \nabla f_x(x)$ and we recall, in the context of f, ∇ is always with respect to the decision argument. Likewise, given $x \in \mathcal{X}$, we use ∇f_x to denote $\nabla f_x(\cdot)$. Note that the continuous differentiability of $f_{x'}(\cdot)$ together with the continuity of $\nabla f_\cdot(x)$ implies ∇f is continuous; the analyses below can be extended to the case where $\nabla f_{x'}(\cdot)$ is a subdifferential.[5] Finally, we assume, $\forall x \in \mathcal{X}$, $\|\nabla f_x(x)\|_* \leq G$ for some $G < \infty$.

Convexity For $\mu \geq 0$, a function $h : \mathcal{X} \to \mathbb{R}$ is called μ-strongly convex, if it satisfies, for all $x, x' \in \mathcal{X}$, $h(x') \geq h(x) + \langle \nabla h(x), x' - x \rangle + \frac{\mu}{2}\|x - x'\|^2$. If h satisfies above with $\mu = 0$, it is called convex. A function h is called pseudo-convex, if $\langle \nabla h(x), x' - x \rangle \geq 0$ implies $h(x') \geq h(x)$. These definitions have a natural inclusion: strongly convex functions are convex; convex functions are pseudoconvex. We say h is L-smooth if ∇h is L-Lipschitz continuous, i.e., there is $L \in [0, \infty)$ such that $\|\nabla h(x) - \nabla h(x')\|_* \leq L\|x - x'\|$ for all $x, x' \in \mathcal{X}$. Finally, we will use Bregman divergence $B_R(x'||x) := R(x') - R(x) - \langle \nabla R(x), x' - x \rangle$ to measure the difference between $x, x' \in \mathcal{X}$, where $R : \mathcal{X} \to \mathbb{R}$ is a μ-strongly convex function with $\mu > 0$; by definition $B_R(\cdot||x)$ is also μ-strongly convex.

Fixed-Point Problems Let $T : \mathcal{X} \to 2^{\mathcal{X}}$ be a point-to-set map, where $2^{\mathcal{X}}$ denotes the power set of $\mathcal{X}$. A fixed-point problem FP$(\mathcal{X}, T)$ aims to find a point $x^\star \in \mathcal{X}$ such that $x^\star \in T(x^\star)$. Suppose T is λ-Lipschitz. It is called non-expansive if $\lambda = 1$, and called λ-contractive if $\lambda < 1$.

Variational Inequalities VIs study equilibriums defined by vector-valued maps. Let $F : \mathcal{X} \to \mathbb{R}^d$ be a point-to-point map. The problems VI$(\mathcal{X}, F)$ and DVI$(\mathcal{X}, F)$ aim to find

[5] Our proof can be extended to upper hemicontinuity for set-valued maps, such as subdifferentials.

$x^\star \in \mathcal{X}$ and $x_\star \in \mathcal{X}$, respectively, such that the following conditions are satisfied:

$$\text{VI} : \langle F(x^\star), x - x^\star \rangle \geq 0, \qquad \forall x \in \mathcal{X}$$
$$\text{DVI} : \langle F(x), x - x_\star \rangle \geq 0, \qquad \forall x \in \mathcal{X}$$

VIs and DVIs are also known as Stampacchia and Minty VIs, respectively (Facchinei and Pang, 2007). The difficulty of solving VIs depends on the property of F. For $\mu \geq 0$, F is called μ-strongly monotone if $\forall x, x' \in \mathcal{X}$. $\langle F(x) - F(x'), x - x' \rangle \geq \mu \|x - x'\|^2$. If F satisfies the above with $\mu = 0$, F is called monotone. F is called pseudo-monotone if $\langle F(x'), x - x' \rangle \geq 0$ implies $\langle F(x), x - x' \rangle \geq 0$ for $x, x' \in \mathcal{X}$. It is known that the gradient of a (strongly/pseudo) convex function is a (strongly/pseudo) monotone.

VIs are generalizations of FPs. For a point-to-point map $T : \mathcal{X} \to \mathcal{X}$, FP$(\mathcal{X}, T)$ is equivalent to VI$(\mathcal{X}, I - T)$, where I is the identity map. If T is λ-contractive, then F is $(1 - \lambda)$-strongly monotone.

Equilibrium Problems EPs further generalize VIs. Let $\Phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a bifunction such that $\Phi(x, x) \geq 0$. The problems EP$(\mathcal{X}, \Phi)$ and DEP$(\mathcal{X}, \Phi)$ aim to find $x^\star, x_\star \in \mathcal{X}$, respectively, s.t.

$$\text{EP} : \Phi(x^\star, x) \geq 0, \qquad \forall x \in \mathcal{X}$$
$$\text{DEP} : \Phi(x, x_\star) \leq 0, \qquad \forall x \in \mathcal{X}.$$

By definition, we have VI$(\mathcal{X}, F) =$ EP$(\mathcal{X}, \Phi)$ if we define $\Phi(x, x') = \langle F(x), x' - x \rangle$.

We can also define monotonicity properties for EPs. For $\mu \geq 0$, Φ is called μ-strongly monotone if for $\forall x, x' \in \mathcal{X}$, $\Phi(x, x') + \Phi(x', x) \leq -\mu \|x - x'\|^2$. It is called monotone, if it satisfies the above with $\mu = 0$. Similarly, Φ is called pseudo-monotone, if $\Phi(x, x') \geq 0$ implies $\Phi(x', x) \leq 0$ for $x, x' \in \mathcal{X}$. One can verify that these definitions are consistent with the ones for VIs.

Primal and Dual Solutions We establish some basics of the solution concepts of EPs. As VIs are a special case of EPs, these results can be applied to VIs too. First, we have a basic relationship between the solution sets, $X^\star$ of EP and $X_\star$ of DEP.

Proposition 8.3.1. (Bianchi and Schaible, 1996) *If Φ is pseudo-monotone, $X^\star \subseteq X_\star$. If $\Phi(\cdot, x)$ is continuous $\forall x \in \mathcal{X}$, $X_\star \subseteq X^\star$.*

The proposition states that a dual solution is always a primal solution when the problem is continuous, and a primal solution is a dual solution when the problem is pseudo-monotone. Intuitively, we can think of the primal solutions $X^\star$ as local solutions, and the dual solutions $X_\star$ as global solutions. In particular for VIs, if F is a gradient of some, even nonconvex, function, any solution in $X_\star$ is a global minimum; any local minimum of a pseudo-convex function is a global minimum (Konnov and Laitinen, 2002).

We note, however, that Proposition 8.3.1 does not directly ensure that the solution sets are non-empty. The existence of primal solutions $X^\star$ has been extensively studied. Here we include a basic result, which is sufficient for the scope of our online learning problems with compact and convex $\mathcal{X}$.

Proposition 8.3.2. (Bianchi and Schaible, 1996) *If $\Phi(x, \cdot)$ is convex and $\Phi(\cdot, x)$ is continuous $\forall x \in \mathcal{X}$, $X^\star$ is non-empty.*

Analogous results have been established for VIs and FPs as well. If F and T are continuous then solutions exist for both VI$(\mathcal{X}, F)$ and FP$(\mathcal{X}, T)$, respectively (Facchinei and Pang, 2007). On the contrary, the existence of dual solutions $X_\star$ is mostly based on assumptions. For example, by Proposition 8.3.1, $X_\star$ is non-empty when the problem is pseudo-monotone. Uniqueness can be established with stronger conditions.

Proposition 8.3.3. (Konnov and Laitinen, 2002) *If the conditions of Proposition 8.3.2 are met and Φ is strongly monotone, then the solution to EP$(\mathcal{X}, \Phi)$ is unique.*

8.4 Equivalence and Hardness

We first ask what extra information the COL formulation entails. We present this result as an equivalence between achieving sublinear dynamic in COL and solving several mathematical programming problems.

Theorem 8.4.1. *Let f be given in Definition 8.1.1. Suppose $f_x(\cdot)$ is convex and continuous. The following problems are equivalent:*

1. *Achieving sublinear dynamic regret w.r.t. f.*
2. *VI$(\mathcal{X}, F)$ where $F(x) = \nabla f_x(x)$.*
3. *EP$(\mathcal{X}, \Phi)$ where $\Phi(x, x') = f_x(x') - f_x(x)$.*
4. *FP$(\mathcal{X}, T)$ where $T(x) = \arg\min_{x' \in X} f_x(x')$.*

Therefore, if there is an algorithm that achieves sublinear dynamic regret that in $poly(d)$, then it solves all PPAD problems in polynomial time.

Theorem 8.4.1 says that, because of the existence of a hidden bifunction, achieving sublinear dynamic regret is essentially equivalent to finding an equilibrium $x^\star \in X^\star$, in which $X^\star$ denotes the set of solutions of the EP/VI/FP problems in Theorem 8.4.1. Therefore, a *necessary* condition for sublinear dynamic regret is that $X^\star$ is non-empty. Fortunately, this is true for our problem definition by Proposition 8.3.2.

Moreover, it suggests that extra structure on COL is necessary for algorithms to achieve sublinear dynamic regret that depends polynomially on d (the dimension of $\mathcal{X}$). The requirement of polynomial dependency is important to properly define the problem. Without it, sublinear dynamic regret can be achieved already (at least asymptotically), e.g. by simply discretizing $\mathcal{X}$ (as $\mathcal{X}$ is compact and ∇f is continuous) albeit with an exponentially large constant.

Due to space limitation, we defer the proof of Theorem 8.4.1 to Section 8.A, along with other proofs for this section. But we highlight the key idea is to prove that the gap function $\rho(x) \coloneqq f_x(x) - \min_{x' \in X} f_x(x')$ can be used as a residual function for the above EP/VI/FP in Theorem 8.4.1. In particular, we note that, for the Φ in Theorem 8.4.1, $\rho(x)$ is equivalent to a residual function $r_{ep}(x) \coloneqq \max_{x' \in \mathcal{X}} -\Phi(x, x')$ used in the EP literature.

Below we discuss sufficient conditions on f based on the equivalence between problems in Theorem 8.4.1, so that the EP/VI/FP in Theorem 8.4.1 becomes better structured and hence allows efficient algorithms.

8.4.1 EP and VI Perspectives

We first discuss some structures on f such that the VI/EP in Theorem 8.4.1 can be efficiently solved. From the literature, we learn that the existence of dual solutions is a common prerequisite to design efficient algorithms (Burachik and Millán, 2019; Dang and Lan, 2015; Konnov, 2007; Lin et al., 2018). For example, convergence guarantees on combined relaxation methods (Konnov, 2007) for VIs rely on the assumption that $X_\star$ is non-empty. Here we discuss some sufficient conditions for non-empty $X_\star$, which by Proposition 8.3.1 and Definition 8.1.1 is a subset of $X^\star$.

By Proposition 8.3.1 and 8.3.2, a sufficient condition for non-empty $X_\star$ is *pseudo-monotonicity* of F or Φ (which we recall is a consequence of monotonicity). For our problem, the dual solutions of the EP and VI are *different*, while their primal solutions X^* are the same.

Proposition 8.4.1. *Let $X_\star$ and $X_{\star\star}$ be the solutions to DVI$(\mathcal{X}, F)$ and DEP$(\mathcal{X}, \Phi)$, respectively, where F and Φ are defined in Theorem 8.4.1. Then $X_{\star\star} \subseteq X_\star$. The converse is true if $f_x(\cdot)$ is linear $\forall x \in \mathcal{X}$.*

Proposition 8.4.1 shows that, for our problem, pseudo-monotonicity of Φ is stronger than that of F. This is intuitive: as the pseudo-monotonicity of Φ implies that there is $x_\star$ such that $f_x(x_\star) \leq f_x(x)$, i.e. a decision argument that is consistently better than the

querying argument under the latter's own question, whereas the pseudo-monotonicity of F merely requires the intersection of the half spaces of $\mathcal{X}$ cut by $\nabla f_x(x)$ to be non-empty. Another sufficient assumption for non-empty $X_\star$ of VIs is that $\mathcal{X}$ is sufficiently strongly convex. This condition has recently been used to show fast convergence of mirror descent and conditional gradient descent (Garber and Hazan, 2015; Veliov and Vuong, 2017). We leave this discussion to Section 8.B.

The above assumptions, however, are sometimes hard to verify for COL. Here we define a subclass of COL and provide constructive (but restrictive) conditions.

Definition 8.4.1. We say a COL problem with f is (α, β)-regular if for some $\alpha, \beta \in [0, \infty)$, $\forall x \in \mathcal{X}$,

1. $f_x(\cdot)$ is a α-strongly convex function.

2. $\nabla f_\cdot(x)$ is a β-Lipschitz continuous map.

We call β the *regularity* constant; for short, we will also say ∇f is β-*regular* and f is (α, β)-regular. We note that β is different from the Lipschitz constant of $\nabla f_x(\cdot)$. The constant β defines the degree of online components; in particular, when $\beta = 0$ the learning problem becomes offline. Based on (α, β)-regularity, we have a sufficient condition to monotonicity.

Proposition 8.4.2. *∇f is $(\alpha - \beta)$-strongly monotone.*

Proposition 8.4.2 shows if $\nabla f_x(\cdot)$ does not change too fast with x, then ∇f is strongly monotone in the sense of VI, implying $X^\star = X_\star$ equal to a singleton (but not necessarily the existence of $X_{\star\star}$). Strong monotoncity also implies fast linear convergence is possible for deterministic feedback (Facchinei and Pang, 2007). When $\alpha = \beta$, it implies at least monotonicity, by which we know $X_\star$ is non-empty.

We emphasize that the condition $\alpha \geq \beta$ is not necessary for monotonicity. The monotonicity condition of ∇f more precisely results from the monotonicity of $\nabla f_\cdot(x')$ and

$\nabla f_x(\cdot)$, as $\langle \nabla f_x(x) - \nabla f_{x'}(x'), x - x' \rangle = \langle \nabla f_x(x) - \nabla f_x(x'), x - x' \rangle + \langle \nabla f_x(x') - \nabla f_{x'}(x'), x - y \rangle$. From this decomposition, we can observe that as long as the sum of $\nabla f_\cdot(x')$ and $\nabla f_x(\cdot)$ is monotone for any $x, x' \in \mathcal{X}$, then ∇f is monotone. In the definition of (α, β)-regular problems, no condition is imposed on $\nabla f_\cdot(x)$, so we need $\alpha \geq \beta$ in Proposition 8.4.2.

8.4.2 Fixed-point Perspective

We can also study the feasibility of sublinear dynamic regret from the perspective of the FP in Theorem 8.4.1. Here again we consider (α, β)-regular problems.

Proposition 8.4.3. *Let $\alpha > 0$. If $\alpha > \beta$, then T is $\frac{\beta}{\alpha}$-contractive; if $\alpha = \beta$, T is non-expansive.*

We see again that the ratio $\frac{\beta}{\alpha}$ plays an important role in rating the difficulty of the problem. When $\alpha > \beta$, an efficient algorithm for obtaining the the fixed point solution is readily available (i.e. by contraction) An alternative interpretation is that x_n^* changes at a slower rate than x_n when $\alpha > \beta$ with respect to $\| \cdot \|$

8.5 Monotone EP as COL

After understanding the structures that determine the difficulty of COL, we describe a converse result of Theorem 8.4.1, which converts monotone EPs into COL.

Theorem 8.5.1. *Let EP$(\mathcal{X}, \Phi)$ be monotone with $\Phi(x, x) = 0$.[6] Consider a COL with $f_x(x') = \Phi(x, x')$. Let $\{x_n\}_{n=1}^N$ be any sequence of decisions and define $\hat{x}_N \coloneqq \frac{1}{N} \sum_{n=1}^N x_n$ It holds that $r_{dep}(\hat{x}_N) \leq \frac{1}{N} \mathrm{regret}_N^s$, where $r_{dep}(x') \coloneqq \max_{x \in \mathcal{X}} \Phi(x, x')$ is the dual residual. The same holds for the best decision in $\{x_n\}_{n=1}^N$.*

Theorem 8.5.1 shows monotone EPs can be solved by achieving sublinear static regret in COL, at least in terms of the dual residual. Below we relate bounds on the dual residual back to the primal residual, which we recall is given as $r_{ep}(x) \coloneqq \max_{x' \in \mathcal{X}} -\Phi(x, x')$.

[6] $\Phi(x, x) = 0$ is not a restriction; see Section 8.C.

Theorem 8.5.2. *Suppose $\Phi(\cdot, x)$ is L-Lipschitz, $\forall x \in \mathcal{X}$. If Φ satisfies $\Phi(x, x') = -\Phi(x', x)$, i.e. skew-symmetric, then $r_{ep}(x) = r_{dep}(x)$. Otherwise,*

1. *For $x \in \mathcal{X}$ such that $r_{dep}(x) \leq 2LD_{\mathcal{X}}$, it holds $r_{ep}(x) \leq 2\sqrt{2LD_{\mathcal{X}}}\sqrt{r_{dep}(x)}$.*
2. *If $\Phi(x, \cdot)$ is in addition μ-strongly convex with $\mu > 0$, for $x \in \mathcal{X}$ such that $r_{dep}(x) \leq L^2/\mu$, it holds $r_{ep}(x) \leq 2.8(L^2/\mu)^{1/3} r_{dep}(x)^{2/3}$*

We can view the above results as a generalization of the classic reduction from convex optimization and Blackwell approachability to no-regret learning (Abernethy, Bartlett, and Hazan, 2011). Generally, the rate of primal residual converges slower than the dual residual. However, when the problem is skew-symmetric (which is true for EPs coming from optimization and saddle-point problems; see Section 8.C), we recover the classic results. In this case, we can show $r_{ep}(\hat{x}_N) = r_{dep}(\hat{x}_N) \leq \frac{1}{N}\mathrm{regret}_N^s \leq \frac{1}{N}\mathrm{regret}_N^d = \frac{1}{N}\sum_{n=1}^N r_{ep}(x_n)$.

These results complement the discussion in Section 8.4.1, as monotonicity implies the dual solution set $X_{\star\star}$ is non-empty. Namely, these monotone EPs constitute a class of source problems of COL for which efficient algorithms are available. Proofs and further discussions of this reduction are given in Section 8.C.

8.6 Reduction by Regularity

Inspired by Theorem 8.4.1, we present a reduction from minimizing dynamic regret to minimizing static regret and convergence to $X^\star$. Intuitively, this is possible, because Theorem 8.4.1 suggests achieving sublinear dynamic regret should not be harder than finding $x^\star \in X^\star$. Define $\mathrm{regret}_N^s(x^\star) \coloneqq \sum_{n=1}^N l_n(x_n) - l_n(x^\star) \leq \mathrm{regret}_N^s$.

Theorem 8.6.1. *Let $x^\star \in X^\star$ and $\Delta_n \coloneqq \|x_n - x^\star\|$. If f is (α, β)-regular for $\alpha, \beta \in [0, \infty)$, then for all N,*

$$\mathrm{regret}_N^d \leq \min\{G\textstyle\sum_{n=1}^N \Delta_n, \mathrm{regret}_N^s(x^\star)\} + \sum_{n=1}^N \min\{\beta D_{\mathcal{X}}\Delta_n, \frac{\beta^2}{2\alpha}\Delta_n^2\}$$

If further $X_{\star\star}$ of the dual EP is non-empty, $\mathrm{regret}_N^d \geq \frac{\alpha}{2}\sum_{n=1}^N \|x_n^* - x_\star\|^2$, *where* $x_\star \in X_{\star\star} \subseteq X^\star$.

Theorem 8.6.1 roughly shows that when $x^\star$ exists (e.g. given by the sufficient conditions in the previous section), it provides a stabilizing effect to the problem, so the dynamic regret behaves almost like the static regret when the decisions are around $x^\star$.

This relationship can be used as a powerful tool for understanding the dynamic regret of existing algorithms designed for EPs, VIs, and FPs. These include, e.g., mirror descent (Beck and Teboulle, 2003), mirror-prox (Juditsky, Nemirovski, and Tauvel, 2011; Nemirovski, 2004), conditional gradient descent (Jaggi, 2013), Mann iteration (Mann, 1953), etc. Interestingly, many of those are also standard tools in online learning, with static regret bounds that are well known (Hazan, 2016).

We can apply Theorem 8.6.1 in different ways, depending on the known convergence of an algorithm. For algorithms whose convergence rate of Δ_n to zero is known, Theorem 8.6.1 essentially shows that their dynamic regret is at most $O(\sum_{n=1}^N \Delta_n)$. For the algorithms with only known static regret bounds, we can use a corollary.

Corollary 8.6.1. *If f is (α,β)-regular and $\alpha > \beta$, it holds* $\mathrm{regret}_N^d \leq \mathrm{regret}_N^s(x^\star) + \frac{\beta^2 \widetilde{\mathrm{regret}}_N^s(x^\star)}{2\alpha(\alpha-\beta)}$, *where* $\widetilde{\mathrm{regret}}_N^s(x^\star)$ *denotes the static regret of the linear online learning problem with* $l_n(x) = \langle \nabla f_n(x_n), x\rangle$.

The purpose of Corollary 8.6.1 is not to give a tight bound, but to show that for nicer problems with $\alpha > \beta$, achieving sublinear dynamic regret is not harder than achieving sublinear static regret. For tighter bounds, we still refer to Theorem 8.6.1 to leverage the equilibrium convergence. We note that the results in Section 8.5 and here concern different classes of COL in general, because $\alpha > \beta$ does not necessarily imply the EP$(\mathcal{X},\Phi)$ is monotone, but only VI$(\mathcal{X},F)$ unless $f_x(\cdot)$ is linear.

Finally, we remark Theorem 8.6.1 is directly applicable to expected dynamic regret (the right-hand side of the inequality will be replaced by its expectation) when the learner only

has access to stochastic feedback, because the COL setup in non-anticipating. Similarly, high-probability bounds can be obtained based on martingale convergence theorems, as in (Cesa-Bianchi, Conconi, and Gentile, 2004). In these cases, we note that the regret is defined with respect to l_n in COL, *not* the sampled losses.

8.6.1 Example Algorithms

We showcase applications of Theorem 8.6.1. These bounds are *non-asymptotic* and depend polynomially on d. And the algorithms do not need to know α and β, except to set the stepsize upper bound for first-order methods. Please refer to Section 8.D for the proofs.

Functional Feedback

We first consider the simple greedy update, which sets $x_{n+1} = \arg\min_{x \in X} l_n(x)$. By Proposition 8.4.3 and Theorem 8.6.1, we see that if $\alpha > \beta$, it has $\mathrm{regret}_N^d = O(1)$. For $\alpha = \beta$, we can use algorithms for non-expansive fixed-point problems (Mann, 1953).

Proposition 8.6.1. *For $\alpha = \beta$, there is an algorithm that achieves sublinear dynamic regret in $poly(d)$.*

Exact First-order Feedback

Next we use the reduction in Theorem 8.6.1 to derive dynamic regret bounds for mirror descent, under deterministic first-order feedback. We recall that mirror descent with step size η_n follows

$$x_{n+1} = \arg\min_{x \in \mathcal{X}} \langle \eta_n g_n, x \rangle + B_R(x \| x_n). \tag{8.4}$$

where g_n is feedback direction, B_R is a Bregman divergence with respect to some 1-strongly convex function R. Here we assume additionally that $f_x(\cdot)$ is γ-smooth with $\gamma > 0$ for all $x \in \mathcal{X}$.

Proposition 8.6.2. *Let f be (α, β)-regular and $f_x(\cdot)$ be γ-smooth, $\forall x \in \mathcal{X}$. Let R be 1-strongly convex and L-smooth. If $\alpha > \beta$, $g_n = \nabla l_n(x_n)$, and $\eta_n < \frac{2(\alpha-\beta)}{L(\gamma+\beta)^2}$, then, for some $0 < \nu < 1$, $\text{regret}_N^d \leq (G + \beta D_{\mathcal{X}})\sqrt{2B_R(x^\star \| x_1)} \sum_{n=1}^N \nu^{n-1} = O(1)$ for* (8.4)*.*

Stochastic & Adversarial Feedback

We now consider stochastic and adversarial cases in COL. As discussed, these are directly handled in the feedback, while the (expected) regret is still measured against the true underlying bifunction. Importantly, we make the subtle assumption that bifunction f is fixed before learning. We consider mirror descent in (8.4) with additive stochastic and adversarial feedback given as $g_n = \nabla l_n(x_n) + \epsilon_n + \xi_n$, where $\epsilon_n \in \mathbb{R}^d$ is zero-mean noise with $\mathbb{E}\left[\|\epsilon_n\|_*^2\right] < \infty$ and $\xi_n \in \mathbb{R}^d$ is a bounded adversarial bias. The component ϵ_n can come from observing a stochastic loss $l_n(x; \zeta_n)$ with random variable ζ_n, when the true loss is $l_n(x) = \mathbb{E}_{\zeta_n}[l_n(x; \zeta_n)]$ (i.e. $\nabla l_n(x_n; \zeta_n) = \nabla l_n(x_n) + \epsilon_n$). On the other hand the adversarial component ξ_n can describe extra bias in computation. We consider the expected dynamic regret $\mathbb{E}[\text{regret}_N^d] = \mathbb{E}[\sum_{n=1}^N l_n(x_n) - \min_{x \in \mathcal{X}} l_n(x)]$, where the expectation is over ϵ_n. Define $\Xi \coloneqq \sum_{n=1}^N \|\xi_n\|_*$. By reduction to static regret in Corollary 8.E.1, we have the following proposition.

Proposition 8.6.3. *If f is fixed before learning, $\alpha > \beta$ and $\eta_n = \frac{1}{\sqrt{n}}$, then mirror descent with $g_n = \nabla l_n(x_n) + \epsilon_n + \xi_n$ has $\mathbb{E}[\text{regret}_N^d] = O(\sqrt{N} + \Xi)$.*

8.6.2 Remark

Essentially, our finding indicates that the feasibility of sublinear dynamic regret is related to a problem's properties. For example, the difficulty of the problem depends largely on the ratio $\frac{\beta}{\alpha}$ when there is no other directional information about $\nabla f_\cdot(x)$, such as monotonicity. We have shown when $\beta \leq \alpha$ efficient algorithms are possible. But, for $\beta > \alpha$, we are not aware of any efficient algorithm. If one exists, it would solve all (α, β)-regular problems, which, in turn, would efficiently solve all EP/VI/FP problems as we can formulate them into

the problem of solving COL problems with sublinear dynamic regret by Theorem 8.4.1.

8.7 Extensions

The framework of COL reveals some core properties of dynamic regret. However, while we allow adversary in feedback, it still assumes that the same loss function $f_x(\cdot)$ must be returned by the bifunction for the same query argument $x \in \mathcal{X}$. Therefore, it does not capture some time-varying situations, in which the opponent's strategy can change across rounds. Also, this constraint allows the learner to potentially enumerate the opponent. Here we relax (8.3) and define a generalization of COL. The proofs of this section are included in Section 8.E.

Definition 8.7.1. We say an online learning problem is (α, β)-*predictable* with $\alpha, \beta \in [0, \infty)$ if $\forall x \in \mathcal{X}$,

1. $l_n(\cdot)$ is a α-strongly convex function.
2. $\|\nabla l_n(x) - \nabla l_{n-1}(x)\|_* \leq \beta \|x_n - x_{n-1}\| + a_n$, where $a_n \in [0, \infty)$ and $\sum_{n=1}^{N} a_n = A_N = o(N)$.

These problems generalize COL along two directions: 1) it makes the problem non-stationary 2) it allows adversarial components within a sublinear budget inside the loss function. We note that the second condition above is different from having adversarial feedback, e.g., in Section 8.6.1, because the regret now is measured with respect to the adversarial loss as opposed to those generated by a fixed bifunction. This new condition can make sublinear dynamic regret considerably harder.

Let us further discuss the relationship between (α, β)-predictable and (α, β)-regular problems. First, a contraction property like Proposition 8.4.3 still holds.

Proposition 8.7.1. *For (α, β)-predictable problems with $\alpha > 0$, $\|x_n^* - x_{n-1}^*\| \leq \frac{\beta}{\alpha}\|x_n - x_{n-1}\| + \frac{a_n}{\alpha}$.*

Proposition 8.7.1 shows that when functional feedback is available and $\frac{\beta}{\alpha} < 1$, sublinear dynamic regret can be achieved, e.g., by a greedy update. However, one fundamental difference between predictable problems and continuous problems is the lack of equilibria X^*, which is the foundation of the reduction in Theorem 8.6.1. This makes achieving sublinear dynamic regret much harder when functional feedback is unavailable or when $\alpha = \beta$. Using Proposition 8.7.1, we establish some preliminary results below.

Theorem 8.7.1. *Let $\frac{\beta}{\alpha} < \frac{\alpha}{2L^2\gamma}$. For (α, β)-predictable problems, if $l_n(\cdot)$ is γ-smooth and R is 1-strongly convex and L-smooth, then mirror descent with deterministic feedback and step size $\eta = \frac{\alpha}{2L\gamma^2}$ achieves $\operatorname{regret}_N^d = O(1 + A_N + \sqrt{NA_N})$.*

We find that, in Theorem 8.7.1, mirror descent must maintain a sufficiently large step size in predictable problems, unlike COL problems which allow for decaying step size. When $\alpha = \beta$, we can show that sublinear dynamic regret is possible under functional feedback.

Theorem 8.7.2. *For $\alpha = \beta$, if $A_\infty < \infty$ and $\|\cdot\|$ is the Euclidean norm, then there is an algorithm with functional feedback achieving sublinear dynamic regret. For $d = 1$ and $a_n = 0$ for all n, sublinear dynamic regret is possible regardless of α, β.*

We do not know, however, whether sublinear dynamic regret is feasible when $\alpha = \beta$ and $A_\infty = \infty$. We conjecture this is infeasible when the feedback is only first-order, as mirror descent is insufficient to solve monotone problems using the last iterate (Facchinei and Pang, 2007) which contain COL with $\alpha = \beta$ (a simpler case than predictable online learning with $\alpha = \beta$).

8.8 Conclusion

We present COL, a new class of problems where the gradient varies continuously across rounds with respect to the learner's decisions. We show that this setting can be equated with certain equilibrium problems (EPs). Leveraging this insight, we present a reduction from

monotone EPs to COL, and show necessary conditions and sufficient conditions for achieving sublinear dynamic regret. Furthermore, we show a reduction from dynamic regret to static regret and the convergence to equilibrium points.

There are several directions for future research on this topic. Our current analyses focus on classical algorithms in online learning. We suspect that the use of adaptive or optimistic methods can accelerate convergence to equilibria, if some coarse model can be estimated. In addition, although we present some preliminary results showing the possibility for interpretable dynamic regret rates in predictable online learning, further refinement and understanding the corresponding lower bounds remain important future work. Lastly, while the current formulations restrict the loss to be determined solely by the learner's current decision, extending the discussion to history-dependent bifunctions is an interesting topic.

8.A Complete Proofs of Section 8.4

8.A.1 Proof of Theorem 8.4.1

Highlight

The key idea to proving Theorem 8.4.1 is that the gap function $\rho(x) \coloneqq f_x(x) - \min_{x' \in X} f_x(x')$ can be used as a residual function for the above EP/VI/FP in Theorem 8.4.1. That is, $\rho(x)$ is non-negative, computable in polynomial time (it is a convex program), and $\rho(x) = 0$ if and only if $x \in X^\star$ (because $f_x(\cdot)$ is convex $\forall x \in \mathcal{X}$). Therefore, to show Theorem 8.4.1, we only need to prove that solving one of these problems is equivalent to achieving sublinear dynamic regret.

First, suppose an algorithm generates a sequence $\{x_n \in \mathcal{X}\}$ such that $\lim_{n\to\infty} x_n = x^\star$, for some $x^\star \in X^\star$. To show this implies $\{x_n \in \mathcal{X}\}$ has sublinear dynamic regret, we first show $\lim_{x \to x^\star \in X^\star} \rho(x) = 0$. Then define $\rho_n = \rho(x_n)$. Because $\lim_{n\to\infty} \rho_n = 0$, we have $\mathrm{regret}_N^d = \sum_{n=1}^N \rho_n = o(N)$.

Next, we prove the opposite direction. Suppose an algorithm generates a sequence $\{x_n \in \mathcal{X}\}$ with sublinear dynamic regret. This implies that $\hat{\rho}_N \coloneqq \min_n \rho_n \leq \frac{1}{N}\sum_{n=1}^{N} \rho_n$ is in $o(1)$ and non-increasing. Thus, $\lim_{N\to\infty} \hat{\rho}_N = 0$. As ρ is a proper residual, the algorithm solves the EP/VI/FP problem by returning the decision associated with $\hat{\rho}_N$.

The proof of PPAD-completeness is based on converting the residual of a Brouwer's fixed-point problem to a bifunction, and use the solution along with $\hat{\rho}_N$ above as the approximate solution.

Note that the gap function ρ, despite motivated by dynamic regret here, corresponds to a natural gap function $r_{ep}(x) \coloneqq \max_{x'\in\mathcal{X}} -F(x, x')$ used in the EP literature, showing again a close connection between the dynamic regret and the EP in Theorem 8.4.1. Nonetheless, $\rho(x)$ is not conventional for VIs and FPs. Below we relate $\rho(x)$ to some standard residuals of VIs and FPs under a stronger assumption on f.

Proposition 8.A.1. *For $\epsilon > 0$, consider some $x_\epsilon \in \mathcal{X}$ such that $\rho(x_\epsilon) \leq \epsilon$. If $f_{x_\epsilon}(\cdot)$ is α-strongly convex, then $\lim_{\epsilon\to 0} \langle \nabla f_{x_\epsilon}(x_\epsilon), x - x_\epsilon\rangle \geq 0$, $\forall x \in \mathcal{X}$, and $\lim_{\epsilon\to 0} \|x_\epsilon - T(x_\epsilon)\| = 0$.*

Full proof

Now we give the details of the steps above.

We first show the solutions sets of the EP, the VI, and the FP are identical.

- 2. $\implies$ 3.

  Let $x^\star_{\text{VI}} \in \mathcal{X}$ be a solution to VI$(\mathcal{X}, F)$ where $F(x) = \nabla f_x(x)$. That is, for all $x \in \mathcal{X}$, $\langle \nabla f_{x^\star_{\text{VI}}}(x^\star_{\text{VI}}), x - x^\star_{\text{VI}}\rangle \geq 0$. The sufficient first-order condition for optimality implies that $x^\star_{\text{VI}}$ is optimal for $f_{x^\star_{\text{VI}}}$. Therefore, $f_{x^\star_{\text{VI}}}(x^\star_{\text{VI}}) \leq f_{x^\star_{\text{VI}}}(x)$ for all $x \in \mathcal{X}$, meaning that $x^\star_{\text{VI}}$ is also a solution to EP$(\mathcal{X}, \Phi)$ where $\Phi(x, x') = f_x(x') - f_x(x)$.

- 3. $\implies$ 4.

  Let $x^\star_{\text{EP}} \in \mathcal{X}$ be a solution to EP$(\mathcal{X}, \Phi)$. By definition, it satisfies $f_{x^\star_{\text{EP}}}(x^\star_{\text{EP}}) \leq f_{x^\star_{\text{EP}}}(x)$

for all $x \in \mathcal{X}$, which implies $x^\star_{\text{EP}} = \arg\min_{x\in\mathcal{X}} f_{x^\star_{\text{EP}}}(x) = T(x^\star_{\text{EP}})$. Therefore, $x^\star_{\text{EP}}$ is a also solution to FP$(\mathcal{X}, T)$, where $T(x') = \arg\min_{x\in\mathcal{X}} f_{x'}(x)$.

- $4. \implies 2.$

  If $x^\star_{\text{FP}}$ is a solution to FP$(\mathcal{X}, T)$, then $x^\star_{\text{FP}} = \arg\min_{x\in\mathcal{X}} f_{x^\star_{\text{FP}}}(x)$. By the necessary first-order condition for optimality, we have $\langle \nabla f_{x^\star_{\text{FP}}}(x^\star_{\text{FP}})x - x^\star_{\text{FP}}\rangle \geq 0$ for all $x \in \mathcal{X}$. Therefore $x^\star_{\text{FP}}$ is also a solution to VI$(\mathcal{X}, F)$ where $F(x) = \nabla f_x(x)$.

Let $X^\star$ denote their common solution sets. To finish the proof of equivalence in Theorem 8.4.1, we only need to show that converging to $X^\star$ is equivalent to achieving sublinear dynamic regret.

- Suppose there is an algorithm that generates a sequence $\{x_n \in \mathcal{X}\}$ such that $\lim_{n\to\infty} x_n = x^\star$, for some $x^\star \in X^\star$. To show this implies $\{x_n \in \mathcal{X}\}$ has sublinear dynamic regret, we need a continuity lemma.

  Lemma 8.A.1. $\lim_{x\to x^\star\in X^\star} \rho(x) = 0$.

  Proof. Let $\bar{x} \in T(x)$. Using convexity, we can derive that

  $$\begin{aligned}
  &\rho(x) = f_x(x) - f_x(\bar{x}) \leq \langle \nabla f_x(x), x - \bar{x}\rangle \\
  &\leq \langle \nabla f_{x^\star}(x^\star), x - \bar{x}\rangle + \|\nabla f_{x^\star}(x^\star) - \nabla f_x(x)\|_*\|x - \bar{x}\| \\
  &\leq \langle \nabla f_{x^\star}(x^\star), x^\star - \bar{x}\rangle + \|\nabla f_{x^\star}(x^\star)\|_*\|x - x^\star\| + \|\nabla f_{x^\star}(x^\star) - \nabla f_x(x)\|_*\|x - \bar{x}\| \\
  &\leq \|\nabla f_{x^\star}(x^\star)\|_*\|x - x^\star\| + \|\nabla f_{x^\star}(x^\star) - \nabla f_x(x)\|_*\|x - \bar{x}\|
  \end{aligned}$$

  where the second and the third inequalities are due to Cauchy-Schwarz inequality, and the last inequality is due to that $x^\star$ solves VI$(\mathcal{X}, \nabla f)$. By continuity of ∇f, the above upper bound vanishes as $x \to x^\star$. ■

  For short hand, let us define $\rho_n = \rho(x_n)$; we can then write $\text{regret}^d_N = \sum_{n=1}^N \rho_n$. By Lemma 8.A.1, $\lim_{n\to\infty} x = x^\star$ implies that $\lim_{n\to\infty} \rho_n = 0$. Finally, we show by

contradiction that $\lim_{n\to\infty} \rho_n = 0$ implies $\text{regret}_N^d = o(N)$. Suppose the dynamic regret is linear. Then $c > 0$ exists such that there is a subsequence $\{\rho_{n_i}\}$ satisfying $\rho_{n_i} \geq c$ for all n_i. However, this contradicts with $\lim_{n\to\infty} \rho_n = 0$.

- We can also prove the opposite direction. Suppose an algorithm generates a sequence $\{x_n \in \mathcal{X}\}$ with sublinear dynamic regret. This implies that $\hat{\rho}_N \coloneqq \min_n \rho_n \leq \frac{1}{N}\sum_{n=1}^{N} \rho_n$ is in $o(N)$ and non-increasing. Thus, $\lim_{N\to\infty} \hat{\rho}_N = 0$ and the algorithm solves the VI/EP/FP problem because ρ is a residual. Alternatively, we may view $\hat{\rho}$ as a Lyapunov-like function. The sequence of minimizers $\hat{x}_N = \arg\min_{x_n} \rho(x_n)$ are confined to the level sets of ρ, which converge to the zero-level set. Since $\mathcal{X}$ is compact, $\hat{x}_N$ converges to this set.

Finally, we show the PPAD-completeness by proving that achieving sublinear dynamic regret with polynomial dependency on d implies solving a Brouwer's problem (finding a fixed point of a continuous point-to-point map on a convex compact set). Because Brouwer's problem is known to be PPAD-complete Daskalakis, Goldberg, and Papadimitriou, 2009, we can use this algorithm to solve all PPAD problems.

Given a Brouwer's problem on $\mathcal{X}$ with some continuous map $\hat{T}$. We can define the bifunction f as $f_{x'}(x) = \frac{1}{2}\|x - \hat{T}(x')\|_2^2$, where $\|\cdot\|_2$ is Euclidean. Obviously, this f satisfies Definition 8.1.1, and its gap function is zero at $x^\star$ if and only $x^\star$ is a solution to the Brouwer's problem. Suppose we have an algorithm that achieves sublinear dynamic regret for continuous online learning. We can use the definition $\hat{\rho}_N$ in the proof above to return a solution whose gap function is less than $\frac{1}{2}\epsilon^2$, which implies an ϵ-approximate solution to Brouwer's problem (i.e. $\|x - \hat{T}(x)\| \leq \epsilon$). If the dynamic regret depends polynomially on d, we have such an N in $poly(d)$, which implies solving any Brouwer's problem in polynomial time.

Proof of Proposition 8.A.1

For the VI problem, let $x_\epsilon^* = T(x_\epsilon)$ and notice that

$$\frac{\alpha}{2}\|x_\epsilon - x_\epsilon^*\|^2 \leq f_{x_\epsilon}(x_\epsilon) - f_{x_\epsilon}(x_\epsilon^*) \leq \epsilon \tag{8.5}$$

for some $\alpha > 0$. Therefore, for any $x \in \mathcal{X}$,

$$\begin{aligned}
&\langle \nabla f_{x_\epsilon}(x_\epsilon), x - x_\epsilon \rangle \\
&\geq \langle \nabla f_{x_\epsilon}(x_\epsilon^*), x - x_\epsilon \rangle - \|\nabla f_{x_\epsilon}(x_\epsilon^*) - \nabla f_{x_\epsilon}(x_\epsilon)\|_* \|x - x_\epsilon\| \\
&\geq \langle \nabla f_{x_\epsilon}(x_\epsilon^*), x - x_\epsilon^* \rangle - \|\nabla f_{x_\epsilon}(x_\epsilon^*)\|_* \|x_\epsilon - x_\epsilon^*\| - \|\nabla f_{x_\epsilon}(x_\epsilon^*) - \nabla f_{x_\epsilon}(x_\epsilon)\|_* \|x - x_\epsilon\| \\
&\geq -\|\nabla f_{x_\epsilon}(x_\epsilon^*)\|_* \|x_\epsilon - x_\epsilon^*\| - \|\nabla f_{x_\epsilon}(x_\epsilon^*) - \nabla f_{x_\epsilon}(x_\epsilon)\|_* \|x - x_\epsilon\|
\end{aligned}$$

Since $\|x_\epsilon - x_\epsilon^*\|^2 \leq \frac{2\epsilon}{\alpha}$, by continuity of ∇f_{x_ϵ}, it satisfies that $\lim_{\epsilon \to 0} \langle \nabla f_{x_\epsilon}(x_\epsilon), x - x_\epsilon \rangle \geq 0, \forall x \in \mathcal{X}$.

For the fixed-point problem, similarly by (8.5), we see that $\lim_{\epsilon \to 0} \|x_\epsilon - T(x_\epsilon)\| = 0$

8.A.2 Proofs of Proposition 8.4.1

Proof of Proposition 8.4.1. Let $x_\star \in X_{\star\star}$. It holds that $\forall x \in \mathcal{X}$, $0 \geq \Phi(x, x_\star) = f_x(x_\star) - f_x(x) \geq \langle \nabla f_x(x), x_\star - x \rangle$, which implies $x_\star \in X_\star$. The condition for the converse case is obvious. ■

8.A.3 Proof of Proposition 8.4.2

Because ∇f_x is α-strongly monotone, we can derive

$$\begin{aligned}
\langle \nabla f_x(x) - \nabla f_y(y), x - y \rangle &= \langle \nabla f_x(x) - \nabla f_x(y), x - y \rangle + \langle \nabla f_x(y) - \nabla f_y(y), x - y \rangle \\
&\geq \alpha \|x - y\|^2 - \|\nabla f_x(y) - \nabla f_y(y)\|_* \|x - y\| \\
&\geq (\alpha - \beta)\|x - y\|^2
\end{aligned}$$

$\forall x, y \in \mathcal{X}$, where the last step is due to β-regularity.

8.A.4 Proof of Proposition 8.4.3

The result follows immediately from the following lemma.

Lemma 8.A.2. *Suppose f is (α, β)-regular with $\alpha > 0$. Then F in Theorem 8.4.1 is point-valued and $\frac{\beta}{\alpha}$-Lipschitz.*

Proof. Let $x^* = F(x)$ and $y^* = F(y)$ for some $x, y \in \mathcal{X}$. By strong convexity, x^* and y^* are unique, and $\nabla f_x(\cdot)$ is α-strongly monotone; therefore it holds that

$$\begin{aligned}\langle \nabla f_x(y^*), y^* - x^* \rangle &\geq \langle \nabla f_x(x^*), y^* - x^* \rangle + \alpha \|x^* - y^*\|^2 \\ &\geq \alpha \|x^* - y^*\|^2\end{aligned}$$

Since y^* satisfies $\langle \nabla f_y(y^*), x^* - y^* \rangle \geq 0$, the above inequality implies that

$$\begin{aligned}\alpha \|x^* - y^*\|^2 &\leq \langle \nabla f_x(y^*), y^* - x^* \rangle \\ &\leq \langle \nabla f_x(y^*) - \nabla f_y(y^*), y^* - x^* \rangle \\ &\leq \|\nabla f_x(y^*) - \nabla f_y(y^*)\|_* \|y^* - x^*\| \\ &\leq \beta \|x - y\| \|y^* - x^*\|\end{aligned}$$

Rearranging the inequality gives the statement. ■

8.B Dual Solution and Strongly Convex Sets

We show when the strong convexity property of $\mathcal{X}$ implies the existence of dual solution for VIs. We first recall the definition of strongly convex sets.

Definition 8.B.1. Let $\alpha_{\mathcal{X}} \geq 0$. A set $\mathcal{X}$ is called *$\alpha_{\mathcal{X}}$-strongly convex* if, for any $x, x' \in \mathcal{X}$ and $\lambda \in [0, 1]$, it holds that, for all unit vector v, $\lambda x + (1-\lambda)x' + \frac{\alpha_{\mathcal{X}} \lambda(1-\lambda)}{2} \|x - x'\|^2 v \in \mathcal{X}$.

When $\alpha_{\mathcal{X}} = 0$, the definition reduces to usual convexity. Also, we see that this definition implies $\alpha_{\mathcal{X}} \leq \frac{4}{D_{\mathcal{X}}}$. In other words, larger sets are less strongly convex. This can also be seen from the lemma below.

Lemma 8.B.1. (Journée et al., 2010, Theorem 12) *Let f be non-negative, α-strongly convex, and β-smooth on a Euclidean space. Then the set $\{x|f(x) \leq r\}$ is $\frac{\alpha}{\sqrt{2r\beta}}$-strongly convex.*

Here we present the existence result.

Proposition 8.B.1. *Let $x^\star \in X^\star$. If $\mathcal{X}$ is $\alpha_{\mathcal{X}}$-strongly convex $\forall x \in \mathcal{X}$, it holds that*

$$\langle F(x^*), x - x^* \rangle \geq \frac{\alpha_{\mathcal{X}}}{2} \|x - x^*\|^2 \|F(x^*)\|_*$$

If further F is L-Lipschitz, this implies

$$\langle F(x), x - x^* \rangle \geq (\frac{\alpha_{\mathcal{X}}}{2} \|F(x^*)\|_* - L) \|x - x^*\|^2$$

i.e. when $\alpha_{\mathcal{X}} \geq \frac{2L}{\|F(x^*)\|_*}$, $x^\star \in X_\star$.

Proof of Proposition 8.B.1. Let $g = F(x^\star)$. Let $y = \lambda x + (1 - \lambda)x^\star$ and $d = -\lambda(1 - \lambda)\frac{\alpha_{\mathcal{X}}}{2}\|x - y\|^2 v$, for some $\lambda \in [0, 1]$ and some unit vector v to be decided later. By $\alpha_{\mathcal{X}}$-strongly convexity of $\mathcal{X}$, we have $y + d \in \mathcal{X}$. We can derive

$$\begin{aligned}
\langle g, x - x^\star \rangle &= \langle g, x - y - d \rangle + \langle g, y + d - x^\star \rangle \\
&\geq \langle g, x - y \rangle - \langle g, d \rangle \\
&= (1 - \lambda) \langle g, x - x^\star \rangle - \langle g, d \rangle
\end{aligned}$$

which implies $\langle g, x - x^\star \rangle \geq \frac{-\langle g, d \rangle}{\lambda} = (1 - \lambda)\frac{\alpha_{\mathcal{X}}}{2}\|x - x^\star\|^2 \langle g, v \rangle$. Since we are free to choose λ and v, we can set $\lambda = 0$ and $v = \arg\max_{v:\|v\| \leq 1} \langle g, v \rangle$, which yields the inequality in the statement. ■

8.C Complete Proofs of Section 8.5

In this section, we describe a general strategy to reduce monotone equilibrium problems (EPs) to continuous online learning (COL) problems. This reduction can be viewed as refinement and generalization of the classic reduction from convex optimization to adversarial online learning and that from saddle-point problem to two-player adversarial online learning. In comparison, our reduction

1. results in a single-player online learning problem, which allows for unified algorithm design

2. considers potential continuous relationship of the losses between different rounds through the setup of COL, which leads to a predictable online problem amenable to acceleration techniques, such as (Cheng et al., 2019d; Juditsky, Nemirovski, and Tauvel, 2011; Rakhlin and Sridharan, 2012).

3. and extends the concept to general convex problems, namely, monotone EPs, which includes of course convex optimization and convex-concave saddle-point problems but also fixed-point problems (FPs), variational inequalities (VIs), etc.

The results here are summarized as Theorem 8.5.1 and Theorem 8.5.2.

Here we further suppose $\Phi(x, x) = 0$ in the definition of EP. This is not a strong condition. First all the common source problems in introduced below in Section 8.C.1 satisfy this condition. Generally, suppose we have some EP problem with $\Phi'(x, x) > 0$ for some x. We can define $\Phi(x, x) = \Phi'(x, x') - \Phi'(x, x')$. Then the solution of EP$(\mathcal{X}, \Phi)$ are subset of the solution EP$(\mathcal{X}, \Phi')$. In other words, allowing $\Phi(x, x) > 0$ only makes problem easier. We note that the below reduction and discussion can easily be extended to work directly with EPs with $\Phi(x, x) > 0$ by defining instead $f_x(x') = \Phi(x, x') - \Phi(x, x)$, but this will make the presentation less clean.

8.C.1 Background: Equilibrium Problems (EPs)

Let $\mathcal{X}$ be a compact and convex set in a finite dimensional space. Let $F : x \times x' \mapsto \Phi(x, x')$ be a bifunction[7] that is continuous in the first argument, convex in the second argument, and satisfies $\Phi(x, x) = 0$.[8] The problem EP$(\mathcal{X}, F)$ aims to find $x^\star \in \mathcal{X}$ such that

$$\Phi(x^\star, x) \geq 0, \qquad \forall x \in \mathcal{X}$$

Its dual problem DEP$(\mathcal{X}, F)$ finds $x_{\star\star} \in \mathcal{X}$ such that

$$\Phi(x, x_{\star\star}) \leq 0, \qquad \forall x \in \mathcal{X}$$

Based on the problem's definition, a natural residual (or gap function) of EP$(\mathcal{X}, F)$ is

$$r_{ep}(x) \coloneqq -\min_{x' \in \mathcal{X}} \Phi(x, x')$$

which says the degree that the inequality in the EP definition is violated. A residual for DEP$(\mathcal{X}, F)$ can be defined similarly as

$$r_{dep}(x') \coloneqq \max_{x \in \mathcal{X}} \Phi(x, x')$$

Sometimes EPs are called maxInf (or minSup) problems (Jofré and Wets, 2014), because

$$x^\star \in \arg\min_{x \in \mathcal{X}} r_{ep}(x) = \arg\max_{x \in \mathcal{X}} \min_{x' \in \mathcal{X}} \Phi(x, x')$$

In a special case, when $\Phi(\cdot, x)$ is concave. It reduces to a saddle-point problem.

[7] We impose convexity and continuity to simplify the setup; similar results hold for subdifferentials and Lipschitz continuity defined based on hemi-continuity.

[8] As discussed, we concern only EP with $\Phi(x, x) = 0$ here

We say a bifunction F is *monotone* if it satisfies

$$\Phi(x, x') + \Phi(x', x) \leq 0,$$

and we say F is skew-symmetric if

$$\Phi(x, x') = -\Phi(x, x'),$$

which implies F is monotone. Finally, we say the problem EP$(\mathcal{X}, F)$ is monotone, if its bifunction F is monotone.

Examples

We review some source problems of EPs. Please refer to e.g. (Jofré and Wets, 2014; Konnov and Schaible, 2000) for a more complete survey.

Convex Optimization Consider $\min_{x \in \mathcal{X}} h(x)$ where h is convex. We can simply define

$$\Phi(x, x') = h(x') - h(x)$$

which is a skew-symmetric (and therefore monotone) bifunction.

We can also define (following the VI given by its optimality condition)

$$\Phi(x, x') = \langle \nabla h(x), x' - x \rangle .$$

We can easily verify that this construction is also monotone

$$\Phi(x, x') + \Phi(x', x) = \langle \nabla h(x), x' - x \rangle + \langle \nabla h(x'), x - x' \rangle = \langle \nabla h(x) - \nabla h(x'), x' - x \rangle \leq 0.$$

Suppose h is μ-strongly convex. We can also consider

$$\Phi(x, x') = \langle \nabla h(x), x' - x \rangle + \frac{\mu'}{2} \|x' - x\|^2$$

where $\mu' \leq \mu$. Such F is still monotone:

$$\Phi(x, x') + \Phi(x', x) = \langle \nabla h(x) - \nabla h(x'), x' - x \rangle + \mu' \|x' - x\|^2 \leq 0.$$

Saddle-Point Problem Let $\mathcal{U}$ and $\mathcal{V}$ to convex and compact sets in a finite dimensional space. Consider a convex-concave saddle point problem

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \phi(u, v) \tag{8.6}$$

in which ϕ is continuous, $\phi(\cdot, y)$ is convex, and $\phi(x, \cdot)$ is concave. It is well known that in this case

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \phi(u, v) = \max_{v \in \mathcal{V}} \min_{u \in \mathcal{U}} \phi(u, v) =: \phi^\star.$$

We can define a EP by the bifunction

$$\Phi(x, x') := -\phi(u, v') + \phi(u', v). \tag{8.7}$$

By definition we have the skew symmetry property, which implies monotonicity.

Variational Inequality A VI with a vector-valued map F finds $x^\star \in \mathcal{X}$ such that

$$\langle F(x^\star), x - x^\star \rangle \geq 0, \qquad \forall x \in \mathcal{X}.$$

To turn that into a EP, we can simply define

$$\Phi(x, x') = \langle F(x), x' - x \rangle .$$

Mixed Variational Inequality (MVI) MVI considers problems that finds $x^\star \in \mathcal{X}$ such that

$$h(x) - h(x^\star) + \langle F(x^\star), x - x^\star \rangle \geq 0, \qquad \forall x \in \mathcal{X}.$$

Following the previous idea, we can define its equivalent EP through the bifunction

$$\Phi(x, x') = h(x') - h(x) + \langle F(x), x' - x \rangle$$

8.C.2 More insights into residuals of primal and dual EPs

We derive further relationships between primal and dual EPs. These properties will be useful for understanding the reduction introduced in the next section.

Monotonicity

By the definition of monotonicity, $\Phi(x, x') + \Phi(x', x) \leq 0$, we can relate the primal and the dual residuals: for $\hat{x} \in \mathcal{X}$,

$$r_{dep}(\hat{x}) = \max_{x \in \mathcal{X}} \Phi(x, \hat{x}) \leq \max_{x \in \mathcal{X}} -\Phi(\hat{x}, x) = r_{ep}(\hat{x})$$

Let $X^\star$ and $X_{\star\star}$ be the solution sets of the EP and DEP, respectively. In other words, for monotone EPs, $X^\star \subseteq X_{\star\star}$.

Continuity

When $\Phi(\cdot, x)$ is continuous, it can be shown that $X^\star \subseteq X_{\star\star}$ (Konnov and Schaible, 2000) (this can be relaxed to hemi-continuity). Below we relate the primal and the dual residuals in this case. It implies that the convergence rate of the primal residual is slower than the dual residual.

Proposition 8.C.1. *Suppose $\Phi(\cdot, x)$ is L-Lipschitz continuous for any $x \in \mathcal{X}$ and $\max_{x,x' \in \mathcal{X}} \|x - x'\| \leq D$. If $r_{dep}(x) \leq 2LD$, the $r_{ep}(x) \leq 2\sqrt{2LD}\sqrt{r_{dep}(x)}$.*

Suppose in addition $\Phi(x, \cdot)$ is μ-strongly convex with $\mu > 0$. If $r_{dep}(x) \leq \frac{L^2}{\mu}$, we can remove the dependency on D and show $r_{ep}(x) \leq 2.8(\frac{L^2}{\mu})^{1/3} r_{dep}(x)^{2/3}$.

Proof. Let $y \in \mathcal{X}$ be arbitrary. Define $z = \tau x + (1-\tau)y$, where $\tau \in [0, 1]$. Suppose x is an ϵ-approximate dual solution, i.e.,

$$r_{dep}(x) = \max_{x' \in \mathcal{X}} \Phi(x', x) = \epsilon$$

By convexity and $\Phi(z, z) = 0$, we can write

$$\begin{aligned}
\epsilon \geq \Phi(z, x) &= \Phi(z, x) - \Phi(z, z) \\
&\geq \Phi(z, x) - \tau\Phi(z, x) - (1-\tau)\Phi(z, y) = (1-\tau)(\Phi(z, x) - \Phi(z, y))
\end{aligned}$$

Using this, we can then show

$$\begin{aligned}
-\Phi(x, y) &= -\Phi(x, y) + \Phi(z, y) + (\Phi(z, x) - \Phi(z, y)) - \Phi(z, x) + \Phi(x, x) \\
&\leq |\Phi(z, y) - \Phi(x, y)| + |\Phi(x, x) - \Phi(z, x)| + \Phi(z, x) - \Phi(z, y) \\
&\leq 2(1-\tau)L\|x - y\| + \Phi(z, x) - \Phi(z, y) && (\because \text{Lipschitz condition}) \\
&\leq 2(1-\tau)L\|x - y\| + \frac{\epsilon}{1-\tau} && (\because \text{The inequality above}) \\
&\leq 2(1-\tau)LD + \frac{\epsilon}{1-\tau}
\end{aligned}$$

Assume $\epsilon \leq 2LD$ and let $(1-\tau) = \sqrt{\frac{\epsilon}{2LD}}$, which satisfies $\tau \in [0,1]$. Then

$$-\Phi(x,y) \leq 2\sqrt{2LD\epsilon}$$

When we have μ-strong convexity, we have a tighter bound

$$\begin{aligned}\epsilon \geq \Phi(z,x) = \Phi(z,x) - \Phi(z,z) &\geq \Phi(z,x) - \tau\Phi(z,x) - (1-\tau)\Phi(z,y) + \frac{\mu\tau(1-\tau)}{2}\|x-y\|^2 \\ &= (1-\tau)(\Phi(z,x) - \Phi(z,y)) + \frac{\mu\tau(1-\tau)}{2}\|x-y\|^2\end{aligned}$$

Using this, we can instead show (following similar steps as above)

$$\begin{aligned}-\Phi(x,y) &\leq 2(1-\tau)L\|x-y\| + \Phi(z,x) - \Phi(z,y) \\ &\leq 2(1-\tau)L\|x-y\| + \frac{\epsilon}{1-\tau} - \frac{\mu\tau}{2}\|x-y\|^2 \\ &\leq \frac{\epsilon}{1-\tau} + \frac{2L^2(1-\tau)^2}{\mu\tau}\end{aligned}$$

where the last inequality is simply $bx - \frac{a}{2}x^2 \leq \frac{b^2}{2a}$ for $a > 0$. Assume $\epsilon \leq \frac{L^2}{\mu} =: \frac{K}{2}$ and let $(1-\tau) = (\frac{\epsilon}{K})^{1/3} \in [0,1]$. We have the following inequality, where the last step uses $\epsilon \leq \frac{K}{2}$.

$$-\Phi(x,y) \leq \frac{\epsilon}{1-\tau} + \frac{2L^2(1-\tau)^2}{\mu\tau} = \epsilon^{2/3}K^{1/3}\left(1 + \frac{1}{1-(\frac{\epsilon}{K})^{1/3}}\right) \leq 2.2\epsilon^{2/3}K^{1/3}$$

■

Equivalence between primal and dual EPs.

An interesting special case of EP is those with *skew-symmetric* bifunctions, i.e.

$$\Phi(x,x') = -\Phi(x',x)$$

In this case, the EP and the DEP become identical

$$(DEP) \quad \Phi(x, x_{\star\star}) \leq 0 \qquad \Longleftrightarrow \qquad -\Phi(x_{\star\star}, x) \leq 0 \qquad \Longleftrightarrow \qquad \Phi(x_{\star\star}, x) \geq 0 \quad (EP)$$

and we have $X^\star = X_{\star\star}$ and naturally matching residuals

$$r_{dep}(\hat{x}) = r_{ep}(\hat{x}).$$

Recall from the results of the previous two subsections, generally, when $\Phi(\cdot, x)$ is Lipschitz and F is monotone (but not skew-symmetric), we have $X^\star = X_{\star\star}$ (as known before) but only ($\Phi(x, \cdot)$ is convex)

$$r_{dep}(x) \leq r_{ep}(x) \leq \sqrt{2LD}\sqrt{r_{dep}(x)} \tag{8.8}$$

or ($\Phi(x, \cdot)$ is μ-strongly convex)

$$r_{dep}(x) \leq r_{ep}(x) \leq 2.8(\frac{L^2}{\mu})^{1/3} r_{dep}(x)^{2/3}$$

Relationship with VIs

We can reduce a EP into a VI problem. We observe that if a point $x^\star \in \mathcal{X}$ satisfies

$$\Phi(x^\star, x) \geq 0, \qquad \forall x \in \mathcal{X}$$

if only if

$$\nabla_2 \Phi(x^\star, x^\star)^\top (x - x^\star) \geq 0, \qquad \forall x \in \mathcal{X}$$

(i.e. $x^\star$ is a global minimum of the function $\Phi(x^\star, \cdot)$), where ∇_2 denotes the partial derivative with respect to the second argument. Therefore, EP$(\mathcal{X}, \Phi)$ is equivalent to VI$(\mathcal{X}, F)$

$$\text{find } x^\star \in \mathcal{X} \quad \text{s.t.} \qquad \langle F(x), x' - x \rangle \geq 0, \qquad \forall x' \in \mathcal{X}$$

if we define F as

$$F : x \in \mathcal{X} \mapsto F(x) = \nabla_2 \Phi(x, x) \tag{8.9}$$

In a sense, this VI problem is a linearization of the EP problem. In other words, VIs are EPs whose bifunction satisfies that $\Phi(x, \cdot)$ is linear.

By the definition in (8.9), we can show that

$$r_{dvi}(\hat{x}) \leq r_{dep}(\hat{x}) \qquad \text{and} \qquad r_{ep}(\hat{x}) \leq r_{vi}(\hat{x})$$

And if Φ is monotone, then $F = \nabla_2 \Phi(x, x)$ is monotone (though the opposite is not true), because

$$\begin{aligned}
\langle F(x), x' - x \rangle = \langle \nabla_2 \Phi(x, x), x' - x \rangle &\leq \Phi(x, x') && (\because \text{Convexity}) \\
&\leq -\Phi(x', x) && (\because \text{Monotonicity}) \\
&\leq \langle \nabla_2 \Phi(x', x'), x' - x \rangle = \langle F(x'), x' - x \rangle && (\because \text{Convexity})
\end{aligned}$$

Note the converse is not true, unless $\Phi(x, \cdot)$ is linear.

8.C.3 Reduction from Equilibrium Problems to Continuous Online Learning

Now we present the general reduction strategy. Given a EP $(\mathcal{X}, \Phi)$, we propose to define a COL problem by identifying

$$f_x(x') = \Phi(x, x')$$

We can see that this definition is consistent with Theorem 8.4.1: due to $\Phi(x, x) = 0$, it satisfies

$$f_x(x') - f_x(x) = \Phi(x, x') - \Phi(x, x) = \Phi(x, x')$$

Therefore, we can say a COL is *normalized* if $f_x(x) = 0$. In this case, f and Φ are interchangeable.

Below we relate the dynamic regret $\mathrm{regret}_N^d \coloneqq \sum_{n=1}^N f_{x_n}(x_n) - \min_{x\in\mathcal{X}} f_{x_n}(x)$ and the static regret $\mathrm{regret}_N^s \coloneqq \sum_{n=1}^N f_{x_n}(x_n) - \min_{x\in\mathcal{X}} \sum_{n=1}^N f_{x_n}(x)$ of this problem to the convergence to the EP's solution; note that the above definitions use the fact that in COL $l_n(x) = f_{x_n}(x)$.

Dynamic Regret and Primal Residual

We first observe that each instant term in the dynamic regret of this COL problem is exactly the residual function:

$$f_{x_n}(x_n) - \min_{x\in\mathcal{X}} f_{x_n}(x) = -\min_{x\in\mathcal{X}} \Phi(x_n, x) = r_{ep}(x_n)$$

Therefore, the average dynamic regret describes the rate the gap function converges to zero:

$$\sum_{n=1}^N r_{ep}(x_n) = \sum_{n=1}^N f_{x_n}(x_n) - \min_{x\in\mathcal{X}} f_{x_n}(x) = \mathrm{regret}_N^d$$

Note that the above relationship holds also for weighted dynamic regret. In general, it means that if the average dynamic regret converges, then the last iterate must converge to the solution set of the EP (since the residual is non-negative.)

Static Regret and Dual Residual of Monotone EPs

Next we relate the weighted static regret to the dual residual of the EP. Let $\{w_n\}$ be such that $w_n > 0$. Let $\hat{x}_N = \frac{1}{x_{1:N}} \sum_{n=1}^N w_n x_n$ for some $\{x_n \in \mathcal{X}\}_{n=1}^N$, where we define $w_{1:N} \coloneqq \sum_{n=1}^N w_n$. We can derive

$$
\begin{aligned}
r_{dep}(\hat{x}_N) &= \max_{x \in \mathcal{X}} \Phi(x, \hat{x}_N) \\
&\leq \max_{x \in \mathcal{X}} \frac{1}{w_{1:N}} \sum_{n=1}^N w_n \Phi(x, x_n) && (\because \text{Convexity}) \\
&\leq \max_{x \in \mathcal{X}} \frac{1}{w_{1:N}} \sum_{n=1}^N -w_n \Phi(x_n, x) && (\because \text{Monotonocity}) \\
&= -\min_{x \in \mathcal{X}} \frac{1}{w_{1:N}} \sum_{n=1}^N w_n \Phi(x_n, x) \\
&= \frac{1}{w_{1:N}} \sum_{n=1}^N w_n \Phi(x_n, x_n) - \min_{x \in \mathcal{X}} \frac{1}{w_{1:N}} \sum_{n=1}^N w_n \Phi(x_n, x) && (\because \Phi(x_n, x_n) = 0) \\
&= \frac{1}{w_{1:N}} \left(\sum_{n=1}^N w_n f_n(x_n) - \min_{x \in \mathcal{X}} \sum_{n=1}^N w_n f_n(x) \right) \\
&=: \frac{\mathrm{regret}_N^s(w)}{w_{1:N}}
\end{aligned}
$$

Note that the inequality $r_{dep}(\hat{x}_N) \leq \frac{\mathrm{regret}_N^s(w)}{w_{1:N}}$ holds for *any* sequence $\{x_n\}$ and $\{w_n\}$. Interestingly, by (8.8), we see that by the definition of regrets and the property of monotonicity and local Lipschitz continuity, it holds that

$$
\frac{r_{ep}(\hat{x}_N)^2}{2LD} \leq r_{dep}(\hat{x}_N) \leq \frac{\mathrm{regret}_N^s(w)}{w_{1:N}} \leq \frac{\mathrm{regret}_N^d(w)}{w_{1:N}} =: \frac{\sum_{n=1}^N w_n r_{ep}(x_n)}{w_{1:N}}
$$

where L is the Lipschitz constant of $\Phi(\cdot, x)$ and D is the size of $\mathcal{X}$.

8.C.4 Summary

Let us summarize the insights gained from the above discussions.

1. We can reduce EP($\mathcal{X}, \Phi$) with monotone Φ to the COL problem with $l_n(x) = \Phi(x_n, x)$

2. In this COL, the convergence in (weighted) average dynamic regret implies the convergence of the last iterate to the primal solution set. The convergence in (weighted) average static regret implies the convergence of the (weighted) average decision to the dual solution set.

3. Because any dual solution is a primal solution when $\Phi(\cdot, x)$ is continuous, this implies the (weighted) average solution above also converges to the primal solution set. Particularly, if the problem is Lipschitz, we can show $r_{ep} \leq O(\sqrt{r_{dep}})$ and therefore we can also quantify the exact quality of $\hat{x}_N$ in terms of the primal EP (though it results in a slower rate).

4. When the problem is skew-symmetric (as in the case of common reductions from optimization and saddle-point problems), we have exactly $r_{ep} = r_{dep}$. This means the average static regret rate directly implies the quality of $\hat{x}_N$ in terms of the primal residual, *without* rate degradation.

8.D Complete Proofs of Section 8.6

8.D.1 Proof of Theorem 8.6.1

The main idea is based on the decomposition that

$$\text{regret}_N^d = \sum_{n=1}^N f_{x_n}(x_n) - f_{x_n}(x^\star) + \sum_{n=1}^N f_{x_n}(x^\star) - f_{x_n}(x_n^*) \tag{8.10}$$

For the first term, $\sum_{n=1}^{N} f_{x_n}(x_n) - f_{x_n}(x^\star) = \mathrm{regret}_N^s(x^\star) \leq \mathrm{regret}_N^s$ and $f_{x_n}(x_n) - f_{x_n}(x^\star) \leq \langle \nabla f_{x_n}(x_n), x_n - x^\star \rangle \leq G\Delta_n$. For the second term, we derive

$$
\begin{aligned}
& f_{x_n}(x^\star) - f_{x_n}(x_n^*) \\
& \leq \langle \nabla f_{x_n}(x^\star), x^\star - x_n^* \rangle - \frac{\alpha}{2}\|x^\star - x_n^*\|^2 \\
& \leq \langle \nabla f_{x_n}(x^\star) - \nabla_{x^\star} f(x^\star), x^\star - x_n^* \rangle - \frac{\alpha}{2}\|x^\star - x_n^*\|^2 \\
& \leq \|\nabla f_{x_n}(x^\star) - \nabla_{x^\star} f(x^\star)\|_* \|x^\star - x_n^*\| - \frac{\alpha}{2}\|x^\star - x_n^*\|^2 \\
& \leq \beta \|x_n - x^\star\| \|x^\star - x_n^*\| - \frac{\alpha}{2}\|x^\star - x_n^*\|^2 \\
& \leq \min\{\beta D_{\mathcal{X}} \|x_n - x^\star\|, \frac{\beta^2}{2\alpha}\|x_n - x^\star\|^2\}
\end{aligned}
$$

in which the second inequality is due to that $x^\star \in X^\star$ and the fourth inequality is due to β-regularity. Combining the two terms gives the upper bound. For the lower bound, we notice that when $x_\star \in X_\star$, we have $f_{x_n}(x_n) - f_{x_n}(x_\star) \geq 0$. Since by Proposition 8.3.1 $x_\star \in X^\star$ is also true, we can use (8.10) and the fact that $f_{x_n}(x_\star) - f_{x_n}(x_n^*) \geq \frac{\alpha}{2}\|x_\star - x_n^*\|^2$ to derive the lower bound.

8.D.2 Proof of Corollary 8.6.1

By Proposition 8.4.2, ∇f is $(\alpha - \beta)$-strongly monotone, implying $\langle \nabla f_{x_n}(x_n), x_n - x^\star \rangle \geq (\alpha - \beta)\Delta_n^2$, where we recall that $\Delta_n = \|x_n - x^\star\|$ and $x^\star \in X^\star$. Because

$$
\sum_{n=1}^{N} \langle \nabla f_{x_n}(x_n), x_n - x^\star \rangle = \widetilde{\mathrm{regret}}_N^s(x^\star) \leq \widetilde{\mathrm{regret}}_N^s,
$$

we have by Theorem 8.6.1 the inequality in the statement.

8.D.3 Proof of Proposition 8.6.1

In this case, by Proposition 8.4.3, T is non-expansive. We know that, e.g., Mann iteration (Mann, 1953), i.e., for $\eta_n \in (0, 1)$ we set

$$x_{n+1} = \eta_n x_n + (1 - \eta_n) x_n^*, \tag{8.11}$$

converges to some $x^\star \in X^\star$; in view of (8.11), the greedy is update is equivalent to Mann iteration with $\eta_n = 1$. As Mann iteration converges in general Hilbert space, by Theorem 8.4.1, it has sublinear dynamic regret with some constant that is polynomial in d.

8.D.4 Proof of Proposition 8.6.2

We first establish a simple lemma related to the smoothness of $\nabla f_x(x)$ and then a result on the convergence of the Bregman divergence $B_R(x_n \| x^\star)$. The purpose of the second lemma is to establish essentially a contraction showing that the distance between the equilibrium point $x^\star$ and x_n strictly decreases.

Lemma 8.D.1. *If, $\forall x \in \mathcal{X}$, $\nabla f_\cdot(x)$ is β-Lipschitz continuous and $f_x(\cdot)$ is γ-smooth, then, for any $x, y \in \mathcal{X}$,*

$$\|\nabla f_x(x) - \nabla f_y(y)\|_* \leq (\gamma + \beta)\|x - y\|.$$

Proof. For any $x, y \in \mathcal{X}$, it holds that

$$\begin{aligned}
\|\nabla f_x(x) - \nabla f_y(y)\|_* &\leq \|\nabla f_x(x) - \nabla f_y(x) + \nabla f_y(x) - \nabla f_y(y)\|_* \\
&\leq \|\nabla f_x(x) - \nabla f_y(x)\|_* + \|\nabla f_y(x) - \nabla f_y(y)\|_* \\
&\leq \beta\|x - y\| + \gamma\|x - y\|.
\end{aligned}$$

The last inequality uses β-regularity and γ-smoothness of $\nabla f_\cdot(x)$ and $f_y(\cdot)$, respectively.

■

Lemma 8.D.2. *If f is (α, β)-regular, $f_x(\cdot)$ is γ-smooth for all $x \in \mathcal{X}$, and R is 1-strongly convex and L-smooth, then for the online mirror descent algorithm it holds that*

$$B_R(x^\star \| x_n) \leq \left(1 - 2\eta(\alpha - \beta)L^{-1} + \eta^2(\gamma + \beta)^2\right)^{n-1} B_R(x^\star \| x_1).$$

Proof. By the mirror descent update rule in (8.4), $\langle \eta \nabla f_{x_n}(x_n) + \nabla R(x_{n+1}) - \nabla R(x_n), x^\star - x_{n+1} \rangle \geq 0$. Since $x^\star \in X_\star$, $\langle \eta \nabla f_{x^\star}(x^\star), x_{n+1} - x^\star \rangle \geq 0$. Combining these inequalities yields $\eta \langle \nabla f_{x_n}(x_n) - \nabla f_{x^\star}(x^\star), x_{n+1} - x^\star \rangle \leq \langle \nabla R(x_{n+1}) - \nabla R(x_n), x^\star - x_{n+1} \rangle$. Then by the three-point equality of the Bregman divergence, we have

$$B_R(x^\star \| x_{n+1}) \leq B_R(x^\star \| x_n) - B_R(x_{n+1} \| x_n) - \eta \langle \nabla f_{x_n}(x_n) - \nabla f_{x^\star}(x^\star), x_{n+1} - x^\star \rangle.$$

Because of the $(\alpha - \beta)$-strong monotonicity of $\nabla f_x(x)$, the above inequality implies

$$\begin{aligned}
B_R(x^\star \| x_{n+1}) &\leq B_R(x^\star \| x_n) - B_R(x_{n+1} \| x_n) - \eta \langle \nabla f_{x_n}(x_n) - \nabla f_{x^\star}(x^\star), x_{n+1} - x_n \rangle \\
&\quad - \eta \langle \nabla f_{x_n}(x_n) - \nabla f_{x^\star}(x^\star), x_n - x^\star \rangle \\
&\leq B_R(x^\star \| x_n) - B_R(x_{n+1} \| x_n) - \eta \langle \nabla f_{x_n}(x_n) - \nabla f_{x^\star}(x^\star), x_{n+1} - x_n \rangle \\
&\quad - \eta(\alpha - \beta)\|x^\star - x_n\|^2 \\
&\leq B_R(x^\star \| x_n) + \frac{\eta^2(\gamma + \beta)^2}{2}\|x^\star - x_n\|^2 - \eta(\alpha - \beta)\|x^\star - x_n\|^2 \\
&\leq \left(1 + \eta^2(\gamma + \beta)^2 - 2\eta(\alpha - \beta)L^{-1}\right) B_R(x^\star \| x_n).
\end{aligned}$$

The third inequality results from the Cauchy-Scwharz inequality followed by maximizing over $\|x_{n+1} - x_n\|$ and then applying Lemma 8.D.1. The last inequality uses the fact that R is 1-strongly convex and L-smooth. ■

If $\alpha > \beta$ and η is chosen such that $\eta < \frac{2(\alpha-\beta)}{L(\gamma+\beta)^2}$, we can see that the online mirror descent algorithm guarantees linear convergence of $B_R(x^\star \| x_n)$ to zero with rate $(1 - 2\eta(\alpha -$

$\beta)L^{-1} + \eta^2(\gamma+\beta)^2) \in (0,1)$. By strong convexity, we have,

$$\begin{aligned}\Delta_n = \|x^\star - x_n\| &\leq \sqrt{2B_R(x^\star\|x_n)} \\ &\leq \sqrt{2}\left(1 + \eta^2(\gamma+\beta)^2 - 2\eta(\alpha-\beta)L^{-1}\right)^{\frac{n-1}{2}} B_R(x^\star\|x_0)^{1/2}.\end{aligned}$$

The proposition follows immediately from combining this result and Theorem 8.6.1.

8.D.5 Proof of Proposition 8.6.3

Recall that $g_n = \nabla l_n(x_n) + \epsilon_n + \xi_n$. As discussed previously, we assume there exist constants $0 \leq \sigma, \kappa < \infty$ such that $\mathbb{E}\left[\|\epsilon_n\|_*^2\right] \leq \sigma^2$ and $\|\xi_n\|_*^2 \leq \kappa^2$ for all n. The mirror descent update rule is given by

$$x_{n+1} = \arg\min_{x\in\mathcal{X}} \langle \eta_n g_n, x\rangle + B_R(x\|x_n). \tag{8.12}$$

We use Corollary 8.6.1 along with known results for the static regret to bound the dynamic regret in the stochastic case. The main idea of the proof is to show the result for the linearized losses. By convexity, this can be used to bound both terms in Corollary 8.6.1.

Let u be any fixed vector in $\mathcal{X}$, chosen independent of the learner's decisions $x_1, \ldots, x_n$. The first-order condition for optimality of (8.12) yields $\langle \eta_n g_n, x_{n+1} - u\rangle \leq \langle u - x_{n+1}, \nabla R(x_{n+1}) - \nabla R(x_n)\rangle$. We use this condition to bound the linearized losses as in the proof of Proposition 8.6.2. We can bound the linearized losses by the magnitude of the stochastic gradients and Bregman divergences between u and the learner's decisions:

$$\begin{aligned}\langle g_n, x_n - u\rangle &\leq \frac{1}{\eta_n}\langle u - x_{n+1}, \nabla R(x_{n+1}) - \nabla R(x_n)\rangle + \langle g_n, x_n - x_{n+1}\rangle \\ &= \frac{1}{\eta_n}B_R(u\|x_n) - \frac{1}{\eta_n}B_R(u\|x_{n+1}) - \frac{1}{\eta_n}B_R(x_{n+1}\|x_n) + \langle g_n, x_n - x_{n+1}\rangle \\ &\leq \frac{1}{\eta_n}B_R(u\|x_n) - \frac{1}{\eta_n}B_R(u\|x_{n+1}) - \frac{1}{2\eta_n}\|x_n - x_{n+1}\|^2 + \|g_n\|_*\|x_n - x_{n+1}\| \\ &\leq \frac{1}{\eta_n}B_R(u\|x_n) - \frac{1}{\eta_n}B_R(u\|x_{n+1}) + \frac{\eta_n}{2}\|g_n\|_*^2.\end{aligned}$$

The first inequality follows from adding $\langle g_n, x_n - x_{n+1}\rangle$ to both sides of the inequality from the first-order condition for optimality. The equality uses the three-point equality of the Bregman divergence. The second inequality follows from the Cauchy-Schwarz inequality and the fact that $\frac{1}{2}\|x_n - x_{n+1}\|^2 \leq B_R(x_{n+1}\|x_n)$ due to the 1-strong convexity of R. The last inequality maximizes over $\|x_n - x_{n+1}\|$.

Define $\mathcal{R} = \sup_{w_1, w_2 \in \mathcal{X}} B_R(w_1\|w_2)$, which is bounded. Note that $\mathbb{E}\left[\|g_n\|_*^2\right] \leq 3(G^2 + \sigma^2 + \kappa^2)$. Therefore, summing from $n = 1$ to N, it holds for any $u \in \mathcal{X}$ selected before learning,

$$\mathbb{E}\left[\sum_{n=1}^{N} \langle g_n, x_n - u\rangle\right] \leq \mathbb{E}\left[\sum_{n=1}^{N} \left(\frac{1}{\eta_n} - \frac{1}{\eta_{n-1}}\right)\mathcal{R} + \frac{3}{2}(G^2 + \sigma^2 + \kappa^2)\eta_n\right]$$

After rearrangement, we have

$$\mathbb{E}\left[\sum_{n=1}^{N} \langle \nabla l_n(x_n) + \epsilon_n, x_n - u\rangle\right] \leq \mathbb{E}\left[\sum_{n=1}^{N} \left(\frac{1}{\eta_n} - \frac{1}{\eta_{n-1}}\right)\mathcal{R} + \frac{3}{2}(G^2 + \sigma^2 + \kappa^2)\eta_n + D_{\mathcal{X}}\|\xi_n\|_*\right].$$

Choosing $\eta_n = \frac{1}{\sqrt{n}}$, $\eta_n = \eta_1$, and $u = x^\star$ (because $x^\star$ is fixed for a fixed f selected before learning) yields $\mathbb{E}\left[\sum_{n=1}^{N} \langle \nabla l_n(x_n) + \epsilon_n, x_n - x^\star\rangle\right] = O(\sqrt{N} + \Xi)$. Because of the law of total expectation and that x_n does not depend on ϵ_n, we have $\mathbb{E}[\widetilde{\mathrm{regret}}_N^s(x^\star)] = \mathbb{E}\left[\sum_{n=1}^{N} \langle \nabla l_n(x_n) + \epsilon_n, x_n - x^\star\rangle\right]$. Further, by convexity, it follows $\mathbb{E}[\mathrm{regret}_N^s(x^\star)] \leq \mathbb{E}[\widetilde{\mathrm{regret}}_N^s(x^\star)]$. Then, we may apply Corollary 8.6.1 to obtain the result. Note that there is no requirement that R is smooth.

8.E Complete Proofs of Section 8.7

8.E.1 Proof of Proposition 8.7.1

Because $\nabla l_n(\cdot)$ is α-strongly monotone, it holds

$$\left\langle \nabla l_n(x_{n-1}^*), x_{n-1}^* - x_n^* \right\rangle \geq \alpha \|x_{n-1}^* - x_n^*\|^2$$

Since y^* satisfies $\left\langle \nabla l_{n-1}(x^*_{n-1}), x^*_n - x^*_{n-1} \right\rangle \geq 0$, the above inequality implies that

$$\begin{aligned}\alpha \|x^*_n - x^*_{n-1}\|^2 &\leq \left\langle \nabla l_n(x^*_{n-1}) - \nabla l_{n-1}(x^*_{n-1}), x^*_{n-1} - x^*_n \right\rangle \\ &\leq (\beta \|x_n - x_{n-1}\| + a_n)\|x^*_{n-1} - x^*_n\|\end{aligned}$$

Rearranging the inequality gives the statement.

8.E.2 Proof of Theorem 8.7.1

For convenience, define $\lambda := \frac{\beta}{\alpha}$. Recall that, by the mirror descent update rule, the first-order conditions for optimality of both x_{x+1} and x^*_n yield, for all $x \in \mathcal{X}$,

$$\begin{aligned}\langle \eta \nabla l_n(x_n), x - x_{n+1} \rangle &\geq \langle \nabla R(x_n) - \nabla R(x_{n+1}), x - x_{n+1} \rangle \\ \langle \nabla l_n(x^*_n), x - x^*_n \rangle &\geq 0.\end{aligned}$$

The proof requires many intermediate steps, which we arrange in a series of lemmas that typically follow from each other in order. Ultimately, we aim to achieve a result that resembles a contraction as done in Proposition 8.6.2 but with additional terms due to the adversarial component of the predictable problem. We begin with general bounds on the Bregman divergence beteween the learner's decisions and the optimal decisions.

Lemma 8.E.1. *At round n, for an (α, β)-predictable problem under the mirror descent algorithm, if l_n is γ-smooth and R is 1-strongly convex and L-smooth, then it holds that*

$$\begin{aligned}B_R(x^*_{n+1} \| x_{n+1}) \leq B_R(x^*_{n+1} \| x^*_n) + B_R(x^*_n \| x_{n+1}) \\ + \lambda \|x_{n+1} - x_n\| \|\nabla R(x^*_n) - \nabla R(x_{n+1})\|_* + \frac{a_n}{\alpha} \|\nabla R(x^*_n) - \nabla R(x_{n+1})\|_*\end{aligned}$$

and, in the next round,

$$B_R(x^*_n \| x_{n+1}) \leq B_R(x^*_n \| x_n) - B_R(x_{n+1} \| x_n) - \alpha\eta \|x_n - x^*_n\|^2 + \eta\gamma \|x_n - x^*_n\| \|x_{n+1} - x_n\|.$$

Proof. The first result uses the basic three-point equality of the Bregman divergence followed by the Cauchy-Schwarz inequality and Proposition 8.7.1. Note that this first part of the lemma does not require that x_n is generated from a mirror descent algorithm:

$$\begin{aligned}
B_R(x_{n+1}^*\|x_{n+1}) &= B_R(x_{n+1}^*\|x_n^*) + B_R(x_n^*\|x_{n+1}) + \langle x_{n+1}^* - x_n^*, \nabla R(x_n^*) - \nabla R(x_{n+1})\rangle \\
&\leq B_R(x_{n+1}^*\|x_n^*) + B_R(x_n^*\|x_{n+1}) + \|x_{n+1}^* - x_n^*\|\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_* \\
&\leq B_R(x_{n+1}^*\|x_n^*) + B_R(x_n^*\|x_{n+1}) \\
&\quad + \lambda\|x_{n+1} - x_n\|\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_* + \frac{a_n}{\alpha}\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_*.
\end{aligned}$$

For the second part of the lemma, we require using the first-order conditions of optimality of both x_{n+1} for the mirror descent update and x_n^* for l_n:

$$\begin{aligned}
B_R(x_n^*\|x_{n+1}) &= B_R(x_n^*\|x_n) - B_R(x_{n+1}\|x_n) + \langle x_n^* - x_{n+1}, \nabla R(x_n) - \nabla R(x_{n+1})\rangle \\
&\leq B_R(x_n^*\|x_n) - B(x_{n+1}\|x_n) + \eta\langle \nabla l_n(x_n^*) - \nabla l_n(x_n), x_n - x_n^*\rangle \\
&\quad + \eta\langle \nabla l_n(x_n^*) - \nabla l_n(x_n), x_{n+1} - x_n\rangle \\
&\leq B_R(x_n^*\|x_n) - B_R(x_{n+1}\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \eta\gamma\|x_n - x_n^*\|\|x_{n+1} - x_n\|.
\end{aligned}$$

The first line again applies the three-point equality of the Bregman divergence. The second line combines both first-order optimality conditions to bound the inner product. The last inequality uses the strong convexity of l_n to bound $\eta\langle \nabla l_n(x_n^*) - \nabla l_n(x_n), x_n - x_n^*\rangle \leq -\alpha\eta\|x_n - x_n^*\|^2$ and the Cauchy-Schwarz inequality along with the smoothness of l_n to bound the other inner product. ■

The second result also leads to a natural corollary that will be useful later in the full proof.

Corollary 8.E.1. *Under the same conditions as Lemma 8.E.1, it holds that*

$$B_R(x_n^*\|x_{n+1}) = \left(1 - 2\alpha\eta L^{-1} + \eta^2\gamma^2\right) B_R(x_n^*\|x_n).$$

Proof. We start with the first inequality of Lemma 8.E.1 and then maximize over $\|x_{n+1} - x_n\|^2$. Finally, we applying the strong convexity and smoothness of R to achieve the result:

$$\begin{aligned}
&B_R(x_n^*\|x_{n+1}) \\
&\leq B_R(x_n^*\|x_n) - B_R(x_{n+1}\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \eta\gamma\|x_n - x_n^*\|\|x_{n+1} - x_n\| \\
&\leq (1 - 2\alpha\eta L^{-1})B_R(x_n^*\|x_n) - \frac{1}{2}\|x_{n+1} - x_n\|^2 + \eta\gamma\|x_n - x_n^*\|\|x_{n+1} - x_n\| \\
&\leq (1 - 2\alpha\eta L^{-1})B_R(x_n^*\|x_n) + \eta^2\gamma^2 B_R(x_n^*\|x_n) = \left(1 - 2\alpha\eta L^{-1} + \eta^2\gamma^2\right) B_R(x_n^*\|x_n).\blacksquare
\end{aligned}$$

We can combine both results of Lemma 8.E.1 in order to show

$$\begin{aligned}
&B_R(x_{n+1}^*\|x_{n+1}) \\
&\leq B_R(x_{n+1}^*\|x_n^*) + \lambda\|x_{n+1} - x_n\|\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_* + \frac{a_n}{\alpha}\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_* \\
&\quad + B_R(x_n^*\|x_n) - B(x_{n+1}\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \eta\gamma\|x_n - x_n^*\|\|x_{n+1} - x_n\|.
\end{aligned}$$

Some of the terms in the above inequality can be grouped and bounded above. By L-smoothness of R, we have $B_R(x_{n+1}^*\|x_n^*) \leq \frac{L}{2}\|x_{n+1}^* - x_n^*\|^2 \leq \frac{L}{2}\left(\lambda\|x_n - x_{n+1}\| + \frac{a_n}{\alpha}\right)^2 = \frac{L}{2}\left(\lambda^2\|x_n - x_{n+1}\|^2 + \frac{a_n^2}{\alpha^2} + \frac{2a_n\lambda}{\alpha}\|x_n - x_{n+1}\|\right)$. Because, R is 1-strongly convex, $L \geq 1$; therefore, the previous inequality can be bounded from above using L^2 instead of L. While this artificially worsens the bound, it will be useful for simplifying the conditions sufficient for sublinear dynamic regret. 1-strong convexity of R also gives us $-B_R(x_{n+1}, x_n) \leq -\frac{1}{2}\|x_{n+1} - x_n\|^2$. Applying these upper bounds and then aggregating terms yields

$$\begin{aligned}
&B_R(x_{n+1}^*\|x_{n+1}) \\
&\leq -\frac{(1 - L^2\lambda^2)}{2}\|x_n - x_{n+1}\|^2 + \left(\lambda\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_* + \eta\gamma\|x_n - x_n^*\|\right)\|x_n - x_{n+1}\| \\
&\quad + B_R(x_n^*\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \frac{a_n}{\alpha}\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_* + \frac{a_n^2 L}{2\alpha^2} + \frac{a_n L\lambda}{\alpha}\|x_n - x_{n+1}\| \\
&\leq -\frac{(1 - L^2\lambda^2)}{2}\|x_n - x_{n+1}\|^2 + \left(\lambda\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_* + \eta\gamma\|x_n - x_n^*\|\right)\|x_n - x_{n+1}\|
\end{aligned}$$

$$
\begin{aligned}
&+ B_R(x_n^*\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \frac{a_n L}{\alpha} D_{\mathcal{X}} + \frac{a_n^2 L}{2\alpha^2} + \frac{a_n L\lambda}{\alpha} D_{\mathcal{X}} \\
&\leq \frac{\lambda^2\|\nabla R(x_n^*) - \nabla R(x_{n+1})\|_*^2 + \eta^2\gamma^2\|x_n - x_n^*\|^2}{1 - L^2\lambda^2} + B_R(x_n^*\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \zeta_n \\
&\leq \frac{\lambda^2 L^2\|x_n^* - x_{n+1}\|^2 + \eta^2\gamma^2\|x_n - x_n^*\|^2}{1 - L^2\lambda^2} + B_R(x_n^*\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \zeta_n \\
&\leq \frac{2\lambda^2 L^2 B_R(x_n^*\|x_{n+1}) + 2\eta^2\gamma^2 B_R(x_n^*\|x_n)}{1 - L^2\lambda^2} + B_R(x_n^*\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \zeta_n,
\end{aligned}
$$

where $\zeta_n = \frac{a_n L D_{\mathcal{X}}}{\alpha}(1+\lambda) + \frac{a_n^2 L}{2\alpha^2}$. The third inequality follows from maximizing over $\|x_n - x_{n+1}\|$ and then applying $(a+b)^2 \leq 2a^2 + 2b^2$ for any $a, b \in \mathbb{R}$. For this operation, we require that $L^2\lambda^2 < 1$. The fourth inequality uses L-smoothness of R. The last inequality uses the fact that R is 1-strongly convex to bound the squared normed differences by the Bregman divergence.

We then use Corollary 8.E.1 to bound this result on $B_R(x_{n+1}^*\|x_{n+1})$ in terms of only $B_R(x_n^*\|x_n)$ and the appropriate constants:

$$
\begin{aligned}
&B_R(x_{n+1}^*\|x_{n+1}) \\
&\leq \frac{2L^2\lambda^2 B_R(x_n^*\|x_{n+1}) + 2\eta^2\gamma^2 B_R(x_n^*\|x_n)}{1 - L^2\lambda^2} + B_R(x_n^*\|x_n) - \alpha\eta\|x_n - x_n^*\|^2 + \zeta_n \\
&\leq \frac{2L^2\lambda^2}{1 - L^2\lambda^2}\left(1 - 2\alpha\eta L^{-1} + \eta^2\gamma^2\right) B_R(x_n^*\|x_n) + \frac{2\eta^2\gamma^2}{1 - L^2\lambda^2} B_R(x_n^*\|x_n) \\
&\quad + B_R(x_n^*\|x_n) - 2\alpha\eta L^{-1} B_R(x_n^*\|x_n) + \zeta_n \\
&= \left(1 - 2\alpha\eta L^{-1} + \frac{2\eta^2\gamma^2}{1 - L^2\lambda^2} + \frac{2L^2\lambda^2}{1 - L^2\lambda^2} - \frac{4L\lambda^2\alpha\eta}{1 - L^2\lambda^2} + \frac{2L^2\lambda^2\eta^2\gamma^2}{1 - L^2\lambda^2}\right) B_R(x_n^*\|x_n) + \zeta_n \\
&= \left(\frac{1 + L^2\lambda^2}{1 - L^2\lambda^2}\right)\left(1 - 2\alpha\eta L^{-1} + 2\eta^2\gamma^2\right) B_R(x_n^*\|x_n) + \zeta_n.
\end{aligned}
$$

Thus, we have arrived at an inequality that resembles a contraction. However, the stepsize $\eta > 0$ may be chosen such that it minimizes the factor in front of the Bregman divergence. This can be achieved, but it requires that additional constraints are put on the value of λ.

Lemma 8.E.2. *If $\lambda < \frac{\alpha}{2L^2\gamma}$ and $\eta = \frac{\alpha}{2L\gamma^2}$, then*

$$\left(\frac{1+L^2\lambda^2}{1-L^2\lambda^2}\right)\left(1-2\alpha\eta L^{-1}+2\eta^2\gamma^2\right) < 1$$

Proof. By optimizing over choices of η, it can be seen that

$$1-2\alpha\eta L^{-1}+2\eta^2\gamma^2 \geq 1-\frac{\alpha^2}{2L^2\gamma^2},$$

where η is chosen to be $\frac{\alpha}{2L\gamma^2}$. Therefore, in order to realize a contraction, we must have

$$1 > \left(\frac{1+L^2\lambda^2}{1-L^2\lambda^2}\right)\left(1-\frac{\alpha^2}{2L^2\gamma^2}\right).$$

Alternatively,

$$0 > 2L^2\lambda^2 - \frac{\alpha^2}{2L^2\gamma^2} - \frac{\lambda^2\alpha^2}{2\gamma^2}.$$

The quantity on the right hand size of the above inequality is in fact smaller than $2L^2\lambda^2 - \frac{\alpha^2}{2L^2\gamma^2}$, meaning that it is sufficient to have the condition for a contraction be: $\frac{\alpha}{2L^2\gamma} > \lambda$.. ■

Note that $\frac{\alpha}{2L^2\gamma} < 1$ since $L \geq 1$ and $\gamma \geq \alpha$ by the definitions of smoothness of R and l_n, respectively. Thus, this condition required to guarantee the contraction is stricter than requiring that $\lambda < 1$. If this condition is satisfied and if we set $\eta = \frac{\alpha}{2L\gamma^2}$, then we can further examine the contraction in terms of constants that depend only on the properties of l_n and R:

$$\begin{aligned}
B_R(x^*_{n+1}\|x_{n+1}) &\leq \left(\frac{1+L^2\lambda^2}{1-L^2\lambda^2}\right)\left(1-2\alpha\eta L^{-1}+2\eta^2\gamma^2\right)B_R(x^*_n\|x_n)+\zeta_n \\
&< \left(\frac{1+\frac{\alpha^2}{4L^2\gamma^2}}{1-\frac{\alpha^2}{4L^2\gamma^2}}\right)\left(1-\frac{\alpha^2}{2L^2\gamma^2}\right)B_R(x^*_n\|x_n)+\zeta_n \\
&= \left(1-\frac{\frac{\alpha^4}{8L^4\gamma^4}}{1-\frac{\alpha^2}{4L^2\gamma^2}}\right)B_R(x^*_n\|x_n)+\zeta_n.
\end{aligned}$$

It is easily verified that the factor in front of the Bregman divergence on the right side is less than 1 and greater than $\frac{5}{6}$.

By applying the above inequality recursively, we can derive the inequality below

$$\frac{1}{2}\|x_n - x_n^*\|^2 \leq B_R(x_n^*\|x_n) \leq \rho^{n-1} B_R(x_1^*\|x_1) + \sum_{k=1}^{n-1} \rho^{n-k-1}\zeta_k,$$

where $\rho = \left(\frac{1+L^2\lambda^2}{1-L^2\lambda^2}\right)(1 - 2\alpha\eta L^{-1} + 2\eta^2\gamma^2) < 1$. Therefore the dynamic regret can be bounded as

$$\begin{aligned}
\text{regret}_N^d &= \sum_{n=1}^{N} f_n(x_n) - f_n(x_n^*) \leq G\sum_{n=1}^{N}\|x_n - x_n^*\| \\
&\leq \sqrt{2}GB_R(x_1^*\|x_1)^{1/2}\sum_{n=1}^{N}\rho^{\frac{n-1}{2}} + \sqrt{2}G\sum_{n=2}^{N}\left(\sum_{k=1}^{n-1}\rho^{n-k-1}\zeta_k\right)^{1/2} \\
&\leq \sqrt{2}GB_R(x_1^*\|x_1)^{1/2}\sum_{n=1}^{N}\rho^{\frac{n-1}{2}} + \sqrt{2}G\sum_{n=2}^{N}\sum_{k=1}^{n-1}\rho^{\frac{n-k-1}{2}}\zeta_k^{1/2},
\end{aligned}$$

where both inequalities use the fact that for $a, b > 0$, $a + b \leq a + b + 2\sqrt{ab} = (\sqrt{a} + \sqrt{b})^2$. The left-hand term is clearly bounded above by a constant since $\sqrt{\rho} < 1$. Analysis of the right-hand term is not as obvious, so we establish the following lemma independently.

Lemma 8.E.3. *If $\rho < 1$ and $\zeta_n = \frac{a_n L D_{\mathcal{X}}}{\alpha}(1+\lambda) + \frac{a_n^2 L}{2\alpha^2}$, then it holds that*

$$\sqrt{2}\sum_{n=2}^{N}\sum_{k=1}^{n-1}\rho^{\frac{n-k-1}{2}}\zeta_k^{1/2} = O(A_N + \sqrt{NA_N}).$$

Proof.

$$\sum_{n=2}^{N}\sum_{k=1}^{n-1}\rho^{\frac{n-k-1}{2}}\zeta_k^{1/2} = \sum_{n=1}^{N-1}\zeta_n^{1/2}\left(1 + \rho^{\frac{1}{2}} + \ldots + \rho^{\frac{N-1-n}{2}}\right) \leq \frac{1}{1-\sqrt{\rho}}\sum_{n=1}^{N-1}\sqrt{\zeta_n}.$$

The last inequality upper bounds the finite geometric series with the value of the infinite

geometric series since again $\sqrt{\rho} < 1$ for each k. Recall that ζ_n was defined as

$$\zeta_n = \frac{a_n L D_{\mathcal{X}}}{\alpha}(1+\lambda) + \frac{a_n^2 L}{2\alpha^2}.$$

Therefore, the over the square roots can be bounded:

$$\sum_{n=1}^{N-1}\sqrt{\zeta_n} \leq \sqrt{\frac{LD_{\mathcal{X}}}{\alpha}(1+\lambda)}\sum_{n=1}^{N-1}\sqrt{a_n} + \alpha^{-1}\sqrt{\frac{L}{2}}\sum_{n=1}^{N-1} a_n.$$

While the right-hand summation is simply the definition of A_{N-1}, the left-hand summation yields $\sum_{n=1}^{N-1}\sqrt{a_n} \leq \sqrt{(N-1)A_{N-1}}$. ■

Then the total dynamic regret has order $\text{regret}_N^d = O(1 + A_N + \sqrt{NA_N})$.

8.E.3 Proof of Theorem 8.7.2

Euclidean Space with $\frac{\beta}{\alpha} = 1$

The proof first requires a result from analysis on the convergence of sequences that are nearly monotonic.

Lemma 8.E.4. *Let $(a_n)_{n\in\mathbb{N}} \subset \mathbb{R}$ and $(b_n)_{n\in\mathbb{N}} \subset \mathbb{R}$ be two sequences satisfying $b_n \geq 0$ and $\sum_{k=1}^{n} a_k < \infty\ \forall n \in \mathbb{N}$. If $b_{n+1} \leq b_n + a_n$, then the sequence b_n converges.*

Proof. Define $u_1 := b_1$ and $u_n := b_n - \sum_{k=1}^{n-1} a_k$. Note that $u_1 = b_1 \geq b_2 - a_1 = u_2$. Recursively, $b_n - a_{n-1} \leq b_{n-1} \implies b_n - \sum_{k=1}^{n-1} a_k \leq b_{n-1} - \sum_{k=1}^{n-2}$. Therefore, $u_n \leq u_{n+1}$. Note that $(u_n)_{n\in\mathbb{N}})$ is bounded below because $b_n \geq 0$ and $\sum_{k=1}^{n} a_k < \infty$. This implies that $(u_n)_{n\in\mathbb{N}}$ converges. Because $(\sum_{k=1}^{n} a_k)_{n\in\mathbb{N}}$, also converges, $(b_n)_{n\in\mathbb{N}}$ must converge. ■

The majority of the proof follows a similar line of reasoning as a standard result in the field of discrete-time pursuit-evasion games Alexander, Bishop, and Ghrist, 2006. Let $\|\cdot\|$ denote the Euclidean norm. We aim to show that if the distance between the learner's decision x_n and the optimal decision x_n^* does not converge to zero, then they travel unbounded in a straight line, which is a contradiction.

Consider the following update rule which essentially amounts to a constrained greedy update:

$$x_{n+1} = \frac{x_n + x_n^*}{2}$$

x_{n+1} is well defined at each round because $\mathcal{X}$ is convex. Define $c_n := \|x_n - x_n^*\|$. Then we have

$$\begin{aligned}
0 \leq c_{n+1} &= \|x_{n+1} - x_{n+1}^*\| \\
&\leq \|x_{n+1} - x_n^*\| + \|x_{n+1}^* - x_n^*\| \\
&= \frac{1}{2}\|x_n - x_n^*\| + \|x_{n+1}^* - x_n^*\| \\
&\leq \frac{1}{2}\|x_n - x_n^*\| + \|x_{n+1} - x_n\| + \frac{a_n}{\alpha} \qquad (\because \text{Proposition 8.7.1}) \\
&= \|x_n - x_n^*\| + \frac{a_n}{\alpha} = c_n + \frac{a_n}{\alpha}
\end{aligned}$$

Because it is assumed that $\sum_{n=1}^{\infty} a_n < \infty$, the sequences $(c_n)_{n\in\mathbb{N}}$ and $(a_n)_{n\in\mathbb{N}}$ satisfy the sufficient conditions of Lemma 8.E.4. Thus the sequence $(c_n)_{n\in\mathbb{N}}$ converges, so there exists a limit point $C := \lim_{n\to\infty} c_n \geq 0$. Towards a contradiction, consider the case where $C > 0$. We will prove that this leads the points to follow a straight line in the following lemma.

Lemma 8.E.5. *Let θ_n denote the angle between the vectors from x_n^* to x_{n+1}^* and from x_n^* to x_{n+1}. If $\lim_{n\to\infty} c_n > 0$, then $\lim_{n\to\infty} \cos\theta_n = -1$.*

Proof. At round $n+1$ we can write the distance between the learner's decision and the optimal decision in terms of the previous round:

$$\begin{aligned}
C^2 &= \lim_{n\to\infty} \|x_{n+1} - x_{n+1}^*\|^2 \\
&= \lim_{n\to\infty} \left(\|x_{n+1} - x_n^*\|^2 + \|x_{n+1}^* - x_n^*\|^2 - 2\|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\|\cos\theta_n\right) \\
&\leq \lim_{n\to\infty} \left(\frac{1}{4}\|x_n - x_n^*\|^2 + \|x_n - x_{n+1}\|^2 + \frac{a_n^2}{\alpha^2} + \frac{2a_n}{\alpha}\|x_n - x_{n+1}\| - 2\|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\|\cos\theta_n\right)
\end{aligned}$$

$$
\begin{aligned}
&= \lim_{n\to\infty} \left(\frac{1}{2}\|x_n - x_n^*\|^2 + \frac{a_n^2}{\alpha^2} + \frac{2a_n}{\alpha}\|x_n - x_{n+1}\| - 2\|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\| \cos\theta_n \right) \\
&= \lim_{n\to\infty} \frac{1}{2}\|x_n - x_n^*\|^2 - 2 \lim_{n\to\infty} \|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\| \cos\theta_n \\
&= \frac{1}{2}C^2 - 2 \lim_{n\to\infty} \|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\| \cos\theta_n
\end{aligned}
$$

The first inequality follows because $\|x_{n+1} - x_n^*\| = \frac{1}{2}\|x_n - x_n^*\|$ and $\|x_{n+1}^* - x_n^*\| \leq \|x_{n+1} - x_n\| + \frac{a_n}{\alpha}$ due to Proposition 8.7.1. The next equality again uses $\|x_{n+1} - x_n^*\| = \frac{1}{2}\|x_n - x_n^*\|$. The second to last line follows from passing the limit through the sum, where we have $\lim_{n\to\infty} a_n = 0$ because $A_\infty < \infty$. That is, the inequality above implies

$$
2 \lim_{n\to\infty} \|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\| \cos\theta_n = -\frac{C^2}{2} < 0
$$

which in turn implies $\lim_{n\to\infty} \cos\theta_n < 0$. This leads to an upper bound

$$
\begin{aligned}
&- 2 \lim_{n\to\infty} \|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\| \cos\theta_n \\
&= \left(-2 \lim_{n\to\infty} \cos\theta_n \right) \lim_{n\to\infty} \|x_{n+1} - x_n^*\|\|x_{n+1}^* - x_n^*\| \\
&\leq \left(-2 \lim_{n\to\infty} \cos\theta_n \right) \lim_{n\to\infty} \frac{1}{2}\|x_n - x_n^*\| \left(\|x_{n+1} - x_n\| + \frac{a_n}{\alpha} \right) \\
&= \frac{-C^2}{2} \lim_{n\to\infty} \cos\theta_n
\end{aligned}
$$

Combining these two inequalities, we can then conclude $C^2 \leq \frac{C^2}{2} - \frac{C^2}{2}\cos\theta \leq C^2$. A necessary condition in order for the bounds to be satisfied is $\cos\theta = -1$. ■

When $C > 0$, Lemma 8.E.5 therefore implies the points $x_n, x_{n+1}, x_n^*, x_{n+1}^*$ are colinear in the limit. Thus, $\|x_n - x_{n+m}\|$ grows unbounded in m, which contradicts the compactness of $\mathcal{X}$. The alternative case must then be true: $C = \lim_{n\to\infty} \|x_n - x_n^*\| = 0$. The dynamic regret can then be bounded as:

$$
\text{regret}_N^d = \sum_{n=1}^{N} l_n(x_n) - l_n(x_n^*) \leq G \sum_{n=1}^{N} \|x_n - x_n^*\|
$$

Since $\|x_N - x_N^*\| \to 0$, we know $\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \|x_n - x_n^*\| = 0$. Therefore, the dynamic regret is sublinear.

Note that this result does not reveal a rate of convergence, only that $\|x_n - x_n^*\|$ converges to zero, which is enough for sublinear dynamic regret.

One-dimensional Space with arbitrary $\frac{\beta}{\alpha}$

In the case where $d = 1$, we aim to prove sublinear dynamic regret regardless of α and β by showing that x_n essentially traps x_n^* by taking conservative steps as before. Rather than the constraint being $|x_n - x_{n+1}| \leq \frac{1}{2}|x_n - x_n^*|$, we choose x_{n+1} in the direction of x_n^* subject to $|x_n - x_{n+1}| \leq \frac{1}{1+\lambda}|x_n - x_n^*|$. Specifically, we will use the following update rule:

$$x_{n+1} = \frac{\lambda x_n + x_n^*}{1 + \lambda} \tag{8.13}$$

Recall that sublinear dynamic regret is implied by $c_n := |x_n - x_n^*|$ converging to zero as $n \to \infty$. Therefore, below we will prove the above update rule results in $\lim_{n\to\infty} c_n = 0$. Like our discussions above, this implies achieving sublinear dynamic regret but not directly its rate.

Suppose at any time $|x_n - x_n^*| = 0$. Then we are done since the learner can repeated play the same decision without x_n^* changing. Below we consider the case $|x_n - x_n^*| \neq 0$. We prove this by contradiction. First we observe that the update in (8.13) makes sure that, at any round, x_{n+1}^* cannot switch to the opposite side of x_n^* with respect to x_{n+1} and x_n; namely it is guaranteed that $(x_{n+1}^* - x_{n+1})(x_n^* - x_{n+1}) \geq 0$ and $(x_{n+1}^* - x_n)(x_n^* - x_n) \geq 0$.

Towards a contradiction, suppose that there is some $C > 0$ such that $|x_n - x_n^*| \geq C$ for infinitely many n. Then x_n at every round moves a distance of at least $\frac{C}{1+\lambda}$ in the same direction infinitely since x_{n+1}^* always lies the same side of x_{n+1} as x_n^*. This contradicts the compactness of $\mathcal{X}$. Therefore $|x_n - x_n^*|$ must converge to zero.

CHAPTER 9
A REDUCTION FROM REINFORCEMENT LEARNING TO ONLINE LEARNING

9.1 Introduction

Reinforcement learning (RL) is a fundamental problem for sequential decision making in unknown environments. One of its core difficulties, however, is the need for algorithms to infer long-term consequences based on limited, noisy, short-term feedback. As a result, designing RL algorithms that are both scalable and provably sample efficient has been challenging.

In this chapter, we revisit the classic linear-program (LP) formulation of RL (Denardo and Fox, 1968; Manne, 1959) in an attempt to tackle this long-standing question. We focus on the associated saddle-point problem of the LP (given by Lagrange duality), which has recently gained traction due to its potential for computationally efficient algorithms with theoretical guarantees (Chen, Li, and Wang, 2018; Chen and Wang, 2016; Dai et al., 2018; Lakshminarayanan, Bhatnagar, and Szepesvári, 2018; Lee and He, 2018; Lin, Nadarajah, and Soheili, 2017; Wang, 2017a,b; Wang and Chen, 2016). But in contrast to these previous works based on stochastic approximation, here we consider a reformulation through the lens of online learning, i.e. regret minimization. Since the pioneering work of Gordon (1999) and Zinkevich (2003), online learning has evolved into a ubiquitous tool for systematic design and analysis of iterative algorithms. Therefore, if we can identify a reduction from RL to online learning, we can potentially leverage it to build efficient RL algorithms.

We will show this idea is indeed feasible. We present a reduction by which *any* no-regret online algorithm, after observing N samples, can find a policy $\hat{\pi}_N$ in a policy class Π

satisfying $V^{\hat{\pi}_N}(p) \geq V^{\pi^*}(p) - o(1) - \epsilon_\Pi$, where $V^\pi(p)$ is the accumulated reward of policy π with respect to some initial state distribution p, π^* is the optimal policy, and $\epsilon_\Pi \geq 0$ is a measure of the expressivity of Π (see Section 9.4.2 for definition).

Our reduction is built on a refinement of online learning, called Continuous Online Learning (COL), which was proposed in Chapter 8 to model problems where loss gradients across rounds change continuously with the learner's decisions (Cheng et al., 2019c). As shown in Chapter 8, COL has a strong connection to equilibrium problems (EPs) (Bianchi and Schaible, 1996; Blum, 1994), and any monotone EP (including our saddle-point problem of interest) can be framed as no-regret learning in a properly constructed COL problem (Cheng et al., 2019c). Using this idea, our reduction follows naturally by first converting an RL problem to an EP and then the EP to a COL problem.

Framing RL as COL reveals new insights into the relationship between approximate solutions to the saddle-point problem and approximately optimal policies. Importantly, this new perspective shows that the RL problem can be separated into two parts: regret minimization and function approximation. The first part admits standard treatments from the online learning literature, and the second part can be quantified *independently* of the learning process. For example, one can accelerate learning by adopting optimistic online algorithms (Cheng et al., 2019d; Rakhlin and Sridharan, 2012) (cf. Chapter 10) that account for the predictability in COL, without worrying about function approximators. Because of these problem-agnostic features, the proposed reduction can be used to systematically design efficient RL algorithms with performance guarantees.

As a demonstration, we design an RL algorithm based on arguably the simplest online learning algorithm: mirror descent. Assuming a generative model[1], we prove that, for *any* tabular Markov decision process (MDP), with probability at least $1 - \delta$, this algorithm learns an ϵ-optimal policy for the γ-discounted accumulated reward, using at most $\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}{(1-\gamma)^4\epsilon^2}\right)$ samples, where $|\mathcal{S}|$,$|\mathcal{A}|$ are the sizes of state and action spaces, and γ is

[1] In practice, it can be approximated by running a behavior policy with sufficient exploration (Kearns and Singh, 1999).

the discount rate. Furthermore, thanks to the separation property above, our algorithm admits a natural extension with linearly parameterized function approximators, whose sample and per-round computation complexities become *linear* in the number of parameters, independent of $|\mathcal{S}|$,$|\mathcal{A}|$, though at the cost of policy performance bias due to approximation error.

This sample complexity improves the current best provable rate of the saddle-point RL setup (Chen and Wang, 2016; Lee and He, 2018; Wang, 2017b; Wang and Chen, 2016) by a large factor of $\frac{|\mathcal{S}|^2}{(1-\gamma)^2}$, *without* making any assumption on the MDP.[2] This improvement is attributed to our new online-learning-style analysis that uses a cleverly selected comparator in the regret definition. While it is possible to devise a minor modification of the previous stochastic mirror descent algorithm, e.g. (Wang, 2017b), achieving the same rate with our new analysis, we remark that our algorithm is considerably simpler and removes an unrealistic projection required in previous work (Chen and Wang, 2016; Lee and He, 2018; Wang, 2017b; Wang and Chen, 2016).

Finally, we do note that the same sample complexity can also be achieved, e.g., by model-based RL and (phased) Q-learning (Kakade, 2003; Kearns and Singh, 1999). However, these methods either have super-linear runtime, with no obvious route for improvement, or could become unstable when using function approximators without further assumption. This chapter is partly based on our paper (Cheng et al., 2019a).

Remark 9.1.1. In this chapter, we take a normalized definition of value function V^π (see the section below), which is different from the other chapters of this thesis. In addition, we use F to denote the bifunction in EPs as opposed to Φ in Chapter 8 (we reserve $\mathbf{\Phi}$ to denote the basis functions here). This convention is adopted to making the writing more compact.

[2](Wang, 2017b) has the same sample complexity but requires the MDP to be ergodic under any policy.

9.2 Setup & Preliminaries

Let $\mathcal{S}$ and $\mathcal{A}$ be state and action spaces, which can be discrete or continuous. We consider γ-discounted infinite-horizon problems for $\gamma \in [0, 1)$. Our goal is to find a policy $\pi(a|s)$ that maximizes the discounted average return $V^\pi(p) \coloneqq \mathbb{E}_{s\sim p}[V^\pi(s)]$, where p is the initial state distribution,

$$V^\pi(s) \coloneqq (1-\gamma)\mathbb{E}_{\xi\sim\rho_\pi(s)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right] \tag{9.1}$$

is the value function of π at state s, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function, and $\rho_\pi(s)$ is the distribution of trajectory $\xi = s_0, a_0, s_1, \ldots$ generated by running π from $s_0 = s$ in an MDP. We assume that the initial distribution p, the transition $\mathcal{P}(s'|s, a)$, and the reward function r in the MDP are unknown but can be queried through a *generative model*, i.e. we can sample s_0 from p, s' from $\mathcal{P}$, and $r(s, a)$ for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$. We remark that the definition of V^π in (9.1) contains a $(1-\gamma)$ factor. We adopt this setup to make writing more compact. We denote the optimal policy as π^* and its value function as V^* for short.

9.2.1 Duality in RL

Our reduction is based on the linear-program (LP) formulation of RL. We provide a short recap here (please see Section 9.A and (Puterman, 2014) for details).

To show how $\max_\pi V^\pi(p)$ can be framed as a LP, let us define the average state distribution under π, $d^\pi(s) \coloneqq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_t^\pi(s)$, where d_t^π is the state distribution at time t visited by running π from p (e.g. $d_0^\pi = p$). By construction, d^π satisfies the stationarity property,

$$d^\pi(s') = (1-\gamma)p(s') + \gamma\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}[\mathcal{P}(s'|s, a)]. \tag{9.2}$$

With d^π, we can write $V^\pi(p) = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}\left[r(s, a)\right]$ and our objective $\max_\pi V^\pi(p)$ equiv-

alently as:

$$\begin{aligned} &\max_{\boldsymbol{\mu} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}: \boldsymbol{\mu} \geq \mathbf{0}} \quad \mathbf{r}^\top \boldsymbol{\mu} \\ &\text{s.t.} \quad (1-\gamma)\mathbf{p} + \gamma \mathbf{P}^\top \boldsymbol{\mu} = \mathbf{E}^\top \boldsymbol{\mu} \end{aligned} \tag{9.3}$$

where $\mathbf{r} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $\mathbf{p} \in \mathbb{R}^{|\mathcal{S}|}$, and $\mathbf{P} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|}$ are vector forms of r, p, and $\mathcal{P}$, respectively, and $\mathbf{E} = \mathbf{I} \otimes \mathbf{1} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|}$ (we use $|\cdot|$ to denote the cardinality of a set, $\otimes$ the Kronecker product, $\mathbf{I} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ is the identity, and $\mathbf{1} \in \mathbb{R}^{|\mathcal{A}|}$ the vector of ones). In (9.3), $\mathcal{S}$ and $\mathcal{A}$ are implicitly assumed to have finite cardinalities, but the same formulation extends to countable or even continuous spaces (under proper regularity assumptions; see (Hernández-Lerma and Lasserre, 2012)). We adopt this abuse of notation (emphasized by bold-faced symbols) for compactness.

The variable $\boldsymbol{\mu}$ of the LP in (9.3) resembles a joint distribution $d^\pi(s)\pi(a|s)$. To see this, notice that the constraint in (9.3) is reminiscent of (9.2), and implies $\|\boldsymbol{\mu}\|_1 = 1$, i.e. $\boldsymbol{\mu}$ is a probability distribution. Then one can show $\mu(s,a) = d^\pi(s)\pi(a|s)$ when the constraint is satisfied, which implies that (9.3) is the same as $\max_\pi V^\pi(p)$ and its solution $\boldsymbol{\mu}^*$ corresponds to $\mu^*(s,a) = d^{\pi^*}(s)\pi^*(a|s)$ of the optimal policy π^*.

As (9.3) is a LP, it suggests looking at its dual, which turns out to be the classic LP formulation of RL[3],

$$\begin{aligned} &\min_{\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}} \quad \mathbf{p}^\top \mathbf{v} \\ &\text{s.t.} \quad (1-\gamma)\mathbf{r} + \gamma \mathbf{P}\mathbf{v} \leq \mathbf{E}\mathbf{v}. \end{aligned} \tag{9.4}$$

It can be verified that for all $\mathbf{p} > 0$, the solution to (9.4) satisfies the Bellman equation (Bellman, 1954) and therefore is the optimal value function $\mathbf{v}^*$ (the vector form of V^*). We note

[3]Our setup in (9.4) differs from the classic one in the $(1-\gamma)$ factor in the constraint due to the average setup.

that, for any π, V^π by definition satisfies a stationarity property

$$V^\pi(s) = \mathbb{E}_{a\sim\pi|s}\left[(1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}|s,a}\left[V^\pi(s')\right]\right] \tag{9.5}$$

which can be viewed as a dual equivalent of (9.2) for d^π. Because r is in $[0,1]$, (9.5) implies V^π lies in $[0,1]$ too.

9.2.2 Toward RL: the Saddle-Point Setup

The LP formulations above require knowing the probabilities p and $\mathcal{P}$ and are computationally inefficient. When only generative models are available (as in our setup), one can alternatively exploit the duality relationship between the two LPs in (9.3) and (9.4), and frame RL as a saddle-point problem (Wang and Chen, 2016). Let us define

$$\mathbf{a}_\mathbf{v} \coloneqq \mathbf{r} + \tfrac{1}{1-\gamma}(\gamma\mathbf{P} - \mathbf{E})\mathbf{v} \tag{9.6}$$

as the *advantage function* with respect to $\mathbf{v}$ (where $\mathbf{v}$ is not necessarily a value function). Then the Lagrangian connecting the two LPs can be written as

$$\mathcal{L}(\mathbf{v}, \boldsymbol{\mu}) \coloneqq \mathbf{p}^\top\mathbf{v} + \boldsymbol{\mu}^\top\mathbf{a}_\mathbf{v}, \tag{9.7}$$

which leads to the saddle-point formulation,

$$\min_{\mathbf{v}\in\mathcal{V}}\max_{\boldsymbol{\mu}\in\mathcal{M}}\mathcal{L}(\mathbf{v}, \boldsymbol{\mu}), \tag{9.8}$$

where the constraints are

$$\mathcal{V} = \{\mathbf{v}\in\mathbb{R}^{|\mathcal{S}|} : \mathbf{v}\geq 0, \|\mathbf{v}\|_\infty \leq 1\} \tag{9.9}$$

$$\mathcal{M} = \{\boldsymbol{\mu}\in\mathbb{R}^{|\mathcal{S}||\mathcal{A}|} : \boldsymbol{\mu}\geq 0, \|\boldsymbol{\mu}\|_1 = 1\}. \tag{9.10}$$

The solution to (9.8) is exactly $(\mathbf{v}^*, \boldsymbol{\mu}^*)$, but notice that extra constraints on the norm of $\boldsymbol{\mu}$ and $\mathbf{v}$ are introduced in $\mathcal{V}, \mathcal{M}$, compared with (9.3) and (9.4). This is a common practice, which uses known bound on the solutions of (9.3) and (9.4) (discussed above) to make the search spaces $\mathcal{V}$ and $\mathcal{M}$ in (9.8) compact and as small as possible so that optimization converges faster.

Having compact variable sets allows using first-order stochastic methods, such as stochastic mirror descent and mirror-prox (Juditsky, Nemirovski, and Tauvel, 2011; Nemirovski et al., 2009), to efficiently solve the problem. These methods only require using the generative model to compute unbiased estimates of the gradients $\nabla_{\mathbf{v}}\mathcal{L} = \mathbf{b}_{\boldsymbol{\mu}}$ and $\nabla_{\boldsymbol{\mu}}\mathcal{L} = \mathbf{a}_{\mathbf{v}}$, where we define

$$\mathbf{b}_{\boldsymbol{\mu}} \coloneqq \mathbf{p} + \tfrac{1}{1-\gamma}(\gamma\mathbf{P} - \mathbf{E})^\top \boldsymbol{\mu} \tag{9.11}$$

as the *balance function* with respect to $\boldsymbol{\mu}$. $\mathbf{b}_{\boldsymbol{\mu}}$ measures whether $\boldsymbol{\mu}$ violates the stationarity constraint in (9.3) and can be viewed as the dual of $\mathbf{a}_{\mathbf{v}}$. When the state or action space is too large, one can resort to function approximators to represent $\mathbf{v}$ and $\boldsymbol{\mu}$, which are often realized by linear basis functions for the sake of analysis (Chen, Li, and Wang, 2018).

9.2.3 COL and EPs

Finally, we review the COL setup in (Cheng et al., 2019c), which we will use to design the reduction from the saddle-point problem in (9.8) to online learning in the next section.

Recall that an online learning problem describes the iterative interactions between a learner and an opponent. In round n, the learner chooses a decision x_n from a decision set $\mathcal{X}$, the opponent chooses a per-round loss function $l_n : \mathcal{X} \to \mathbb{R}$ based on the learner's decisions, and then information about l_n (e.g. its gradient $\nabla l_n(x_n)$) is revealed to the learner. The performance of the learner is usually measured in terms of regret with respect

to some $x' \in \mathcal{X}$,

$$\text{regret}_N(x') \coloneqq \sum_{n=1}^N l_n(x_n) - \sum_{n=1}^N l_n(x').$$

When l_n is convex and $\mathcal{X}$ is compact and convex, many no-regret (i.e. $\text{regret}_N(x') = o(N)$) algorithms are available, such as mirror descent and follow-the-regularized-leader (Cesa-Bianchi and Lugosi, 2006; Hazan, 2016; Shalev-Shwartz, 2012).

COL is a subclass of online learning problems where the loss sequence changes continuously with respect to the played decisions of the learner (Cheng et al., 2019c). In COL, the opponent is equipped with a bifunction $f : (x, x') \mapsto f_x(x')$, where any fixed $x' \in \mathcal{X}$, $\nabla f_x(x')$ is continuous in $x \in \mathcal{X}$. The opponent selects per-round losses based on f, but the learner does not know f: in round n, if the learner chooses x_n, the opponent sets

$$l_n(x) = f_{x_n}(x), \tag{9.12}$$

and returns, e.g., a stochastic estimate of $\nabla l_n(x_n)$ (the regret is still measured in terms of the noise-free l_n).

In (Cheng et al., 2019c), a natural connection is shown between COL and equilibrium problems (EPs). As EPs include the saddle-point problem of interest, we can use this idea to turn (9.8) into a COL problem. Recall an EP is defined as follows: Let $\mathcal{X}$ be compact and $F : (x, x') \mapsto F(x, x')$ be a bifunction s.t. $\forall x, x' \in \mathcal{X}$, $F(\cdot, x')$ is continuous, $F(x, \cdot)$ is convex, and $F(x, x) \geq 0$.[4] The problem EP($\mathcal{X}, F$) aims to find $x^\star \in \mathcal{X}$ s.t.

$$F(x^\star, x) \geq 0, \qquad \forall x \in \mathcal{X}. \tag{9.13}$$

By its definition, a natural residual function to quantify the quality of an approximation solution x to EP is $r_{ep}(x) \coloneqq -\min_{x' \in \mathcal{X}} F(x, x')$ which describes the degree to which (9.13)

[4]We restrict ourselves to this convex and continuous case as it is sufficient for our problem setup.

is violated at x. We say a bifunction F is *monotone* if, $\forall x, x' \in \mathcal{X}$, $F(x, x') + F(x', x) \leq 0$, and *skew-symmetric* if the equality holds.

EPs with monotone bifunctions represent general convex problems, including convex optimization problems, saddle-point problems, variational inequalities, etc. For instance, a convex-concave problem $\min_{y \in \mathcal{Y}} \max_{z \in \mathcal{Z}} \phi(y, z)$ can be cast as EP($\mathcal{X}, F$) with $\mathcal{X} = \mathcal{Y} \times \mathcal{Z}$ and the skew-symmetric bifunction (Jofré and Wets, 2014)

$$F(x, x') \coloneqq -\phi(y, z') + \phi(y', z), \tag{9.14}$$

where $x = (y, z)$ and $x' = (y', z')$. In this case, $r_{ep}(x) = \max_{z' \in \mathcal{Z}} \phi(y, z') - \min_{y' \in \mathcal{Y}} \phi(y', z)$ is the duality gap.

Cheng et al. (2019c) show that a learner achieves sublinear dynamic regret in COL if and only if the same algorithm can solve EP($\mathcal{X}, F$) with $F(x, x') = f_x(x') - f_x(x)$. Concretely, they show that, given a monotone EP($\mathcal{X}, F$) with $F(x, x) = 0$ (which is satisfied by (9.14)), one can construct a COL problem by setting $f_{x'}(x) \coloneqq F(x', x)$, i.e. $l_n(x) = F(x_n, x)$. They further show the following:

Proposition 9.2.1. *If F is* skew-symmetric *and $l_n(x) = F(x_n, x)$, then $r_{ep}(\hat{x}_N) \leq \frac{1}{N} \mathrm{regret}_N$, where $\mathrm{regret}_N = \max_{x \in \mathcal{X}} \mathrm{regret}_N(x)$, and $\hat{x}_N = \frac{1}{N} \sum_{n=1}^{N} x_n$; the same guarantee holds also for the best decision in $\{x_n\}_{n=1}^{N}$.*

9.3 An Online Learning View

We present an alternate online-learning perspective on the saddle-point formulation in (9.8). This analysis paves a way for of our reduction in the next section. By reduction, we mean realizing the two steps below:

1. Define a sequence of online losses such that any algorithm with sublinear regret can produce an approximate solution to the saddle-point problem.

2. Convert the approximate solution in the first step to an approximately optimal policy in RL.

Methods to achieve these two steps individually are not new. The reduction from convex-concave problems to no-regret online learning is well known (Abernethy, Bartlett, and Hazan, 2011). Likewise, the relationship between the approximate solution of (9.8) and policy performance is also available; this is how the saddle-point formulation (Wang, 2017b) works in the first place. So couldn't we just use these existing approaches? We argue that purely combining these two techniques fails to fully capture important structure that resides in RL. While this will be made precise in the later analyses, we highlight the main insights here.

Instead of treating (9.8) as an *adversarial* two-player online learning problem (Abernethy, Bartlett, and Hazan, 2011), we adopt the recent reduction to COL (Cheng et al., 2019c) reviewed in Section 9.2.3. The main difference is that the COL approach is based on a single-player setup and retains the Lipschitz continuity in the source saddle-point problem. This single-player perspective is in some sense cleaner and, as we will show in Section 9.4.2, provides a simple setup to analyze effects of function approximators. Additionally, due to continuity, the losses in COL are predictable and therefore make designing fast algorithms possible.

With the help of the COL reformulation, we study the relationship between the approximation solution to (9.8) and the performance of the associated policy in RL. We are able to establish a tight bound between the residual and the performance gap, resulting in a large improvement of $\frac{|\mathcal{S}|^2}{(1-\gamma)^2}$ in sample complexity compared with the best bounds in the literature of the saddle-point setup, *without* adding extra constraints on $\mathcal{X}$ and assumptions on the MDP. Overall, this means that *stronger* sample complexity guarantees can be attained by *simpler* algorithms, as we demonstrate in Section 9.5.

The missing proofs of this section are in Section 9.B.

9.3.1 The COL Formulation of RL

First, let us exercise the above COL idea with the saddle-point formulation of RL in (9.8). To construct the EP, we can let $\mathcal{X} = \{x = (\mathbf{v}, \boldsymbol{\mu}) : \mathbf{v} \in \mathcal{V}, \boldsymbol{\mu} \in \mathcal{M}\}$, which is compact. According to (9.14), the bifunction F of the associated EP($\mathcal{X}, F$) is naturally given as

$$\begin{aligned} F(x, x') &\coloneqq \mathcal{L}(\mathbf{v}', \boldsymbol{\mu}) - \mathcal{L}(\mathbf{v}, \boldsymbol{\mu}') \\ &= \mathbf{p}^\top \mathbf{v}' + \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} - \mathbf{p}^\top \mathbf{v} - \boldsymbol{\mu}'^\top \mathbf{a}_{\mathbf{v}} \end{aligned} \tag{9.15}$$

which is skew-symmetric, and $x^* \coloneqq (\boldsymbol{v}^*, \boldsymbol{\mu}^*)$ is a solution to EP($\mathcal{X}, F$). This identification gives us a COL problem with the loss in the nth round defined as

$$l_n(x) \coloneqq \mathbf{p}^\top \mathbf{v} + \boldsymbol{\mu}_n^\top \mathbf{a}_{\mathbf{v}} - \mathbf{p}^\top \mathbf{v}_n - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}_n} \tag{9.16}$$

where $x_n = (\mathbf{v}_n, \boldsymbol{\mu}_n)$. We see l_n is a linear loss. Moreover, because of the continuity in $\mathcal{L}$, it is predictable, i.e. l_n can be (partially) inferred from past feedback.

9.3.2 Policy Performance

By Proposition 9.2.1, any no-regret algorithm, when applied to (9.16), provides guarantees in terms of the residual function $r_{ep}(x)$ of the EP. But this is not the end of the story. We need to relate the learner decision $x \in \mathcal{X}$ to a policy π in RL and then convert bounds on $r_{ep}(x)$ back to the policy performance $V^\pi(p)$. We follow the common rule in the literature and associate each $x = (\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{X}$ with a policy $\pi_{\boldsymbol{\mu}}$ defined as

$$\pi_{\boldsymbol{\mu}}(a|s) \propto \mu(s, a). \tag{9.17}$$

Below we relate the residual $r_{ep}(x)$ to the performance gap $V^*(p) - V^{\pi_\mu}(p)$. For convenience, we define a relative performance measure for the skew-symmetric bifunction F in

(9.15):

$$r_{ep}(x; x') \coloneqq F(x, x) - F(x, x') = -F(x, x') \tag{9.18}$$

for $x, x' \in \mathcal{X}$. By (9.18), we see that $l_n(x_n) - l_n(x') = r_{ep}(x_n; x')$, so $r_{ep}(x; x')$ compares the performance of x with respect to the comparator x'.

We will discuss upper bounds on the performance gap $V^*(p) - V^{\pi_{\boldsymbol{\mu}}}(p)$ in terms of $r_{ep}(x; x')$, for some properly chosen $x' \in \mathcal{X}$. As $r_{ep}(x; x') \leq \max_{x' \in \mathcal{X}} -F(x, x') = r_{ep}(x)$, these results can translate the convergence of residual $r_{ep}(x)$ (determined by the regret in COL) to the convergence of the policy performance gap. More precisely, we are looking for inequalities in the form $V^*(p) - V^{\pi_{\boldsymbol{\mu}}}(p) \leq \kappa(r_{ep}(x; x'))$ that hold for *all* $x \in \mathcal{X}$ with some strictly increasing function κ and some $x' \in \mathcal{X}$, so we can get *non-asymptotic* performance guarantees at the end once we combine the two steps described at the beginning of this section. That is, by applying results of (Cheng et al., 2019c) to the COL defined in (9.16), we will get $V^*(p) - V^{\hat{\pi}_N}(p) \leq \kappa(\frac{\text{regret}_N(x')}{N})$, where $\hat{\pi}_N$ is the policy associated with the average/best decision.

The Classic Result

Existing approaches (e.g. (Chen and Wang, 2016; Lee and He, 2018; Wang, 2017b)) to the saddle-point point formulation in (9.8) rely on the relative residual $r_{ep}(x; x^*)$ with respect to the optimal solution to the problem x^*, which we restate in our notation.

Proposition 9.3.1. *For any* $x = (\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{X}$, *if* $\mathbf{E}^\top \boldsymbol{\mu} \geq (1 - \gamma)\mathbf{p}$, $r_{ep}(x; x^*) \geq (1 - \gamma) \min_s p(s) \|\mathbf{v}^* - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty$.

Therefore, although the original saddle-point problem in (9.8) is framed using $\mathcal{V}$ and $\mathcal{M}$, in practice, an extra constraint, such as $\mathbf{E}^\top \boldsymbol{\mu} \geq (1 - \gamma)\mathbf{p}$, is added into $\mathcal{M}$, i.e. these

algorithms consider instead

$$\mathcal{M}' = \{\boldsymbol{\mu} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} : \boldsymbol{\mu} \in \mathcal{M}, \mathbf{E}^\top \boldsymbol{\mu} \geq (1-\gamma)\mathbf{p}\}, \tag{9.19}$$

so that the marginal of the estimate $\boldsymbol{\mu}$ can have the sufficient coverage required in Proposition 9.3.1. This condition is needed to establish non-asymptotic guarantees on the performance of the policy generated by $\boldsymbol{\mu}$ (Lee and He, 2018; Wang, 2017b; Wang and Chen, 2016), but it can sometimes be impractical to realize, e.g., when $\mathbf{p}$ is unknown. Without it, extra assumptions (like ergodicity (Wang, 2017b)) on the MDP are needed.

However, Proposition 9.3.1 is undesirable for a number of reasons. First, the bound is quite conservative, as it concerns the uniform error $\|\mathbf{v}^* - \mathbf{v}^{\pi_\mu}\|_\infty$ whereas the objective in RL is about the gap $V^*(p) - V^{\pi_\mu}(p) = \mathbf{p}^\top(\mathbf{v}^* - \mathbf{v}^{\pi_\mu})$ with respect to the initial distribution p (i.e. a weighted error). Second, the constant term $(1-\gamma)\min_s p(s)$ can be quite small (e.g. when p is uniform, it is $\frac{1-\gamma}{|\mathcal{S}|}$) which can significantly amplify the error in the residual. Because a no-regret algorithm typically decreases the residual in $O(N^{-1/2})$ after seeing N samples, the factor of $\frac{1-\gamma}{|\mathcal{S}|}$ earlier would turn into a multiplier of $\frac{|\mathcal{S}|^2}{(1-\gamma)^2}$ in sample complexity. This makes existing saddle-point approaches sample inefficient compared to other RL methods like Q-learning (Kakade, 2003). Lastly, enforcing $\mathbf{E}^\top \boldsymbol{\mu} \geq (1-\gamma)\mathbf{p}$ requires knowing $\mathbf{p}$ (which is unavailable in our setup) and adds extra projection steps during optimization. When $\mathbf{p}$ is unknown, while it is possible to modify this constraint to use a uniform distribution, this might worsen the constant factor and could introduce bias.

One may conjecture that the bound in Proposition 9.3.1 could perhaps be tightened by better analyses. However, we prove this is impossible in general.

Proposition 9.3.2. *There is a class of MDPs such that, for some $x \in \mathcal{X}$, Proposition 9.3.1 is an equality.*

We note that Proposition 9.3.2 does not hold for *all* MDPs. Indeed, if one makes stronger assumptions on the MDP, such as that the Markov chain induced by *every* pol-

icy is ergodic (Wang, 2017b), then it is possible to show, for all $x \in \mathcal{X}$, $r_{ep}(x; x^*) = c\|\mathbf{v}^* - \mathbf{v}^{\pi_\mu}\|_\infty$ for some constant c independent of γ and $|\mathcal{S}|$, when one constrains $\mathbf{E}^\top \boldsymbol{\mu} \geq (1 - \gamma + \gamma\sqrt{c})\mathbf{p}$. Nonetheless, this construct still requires adding an undesirable constraint to $\mathcal{X}$.

Curse of Covariate Shift

Why does this happen? We can view this issue as a form of *covariate shift*, i.e. a mismatch between distributions. To better understand it, we notice a simple equality, which has often been used implicitly, e.g. in the technical proofs of (Wang, 2017b).

Lemma 9.3.1. *For any $x = (\mathbf{v}, \boldsymbol{\mu})$, if $x' \in \mathcal{X}$ satisfies* (9.2) *and* (9.5) *(i.e. $\mathbf{v}'$ and $\boldsymbol{\mu}'$ are the value function and state-action distribution of policy $\pi_{\boldsymbol{\mu}'}$), $r_{ep}(x; x') = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'}$.*

The above implies $r_{ep}(x; x^*) = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*}$, which is non-negative. This term is similar to an equality called the performance difference lemma (Kakade and Langford, 2002; Ng, Harada, and Russell, 1999).

Lemma 9.3.2. *Let $\mathbf{v}^\pi$ and $\boldsymbol{\mu}^\pi$ denote the value and state-action distribution of some policy π. Then for* any *function $\mathbf{v}'$, it holds that $\mathbf{p}^\top(\mathbf{v}^\pi - \mathbf{v}') = (\boldsymbol{\mu}^\pi)^\top \mathbf{a}_{\mathbf{v}'}$. In particular, it implies $V^\pi(p) - V^{\pi'}(p) = (\boldsymbol{\mu}^\pi)^\top \mathbf{a}_{\mathbf{v}^{\pi'}}$.*

From Lemmas 9.3.1 and 9.3.2, we see that the difference between the residual $r_{ep}(x; x^*) = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*}$ and the performance gap $V^{\pi_\mu}(p) - V^{\pi^*}(p) = (\boldsymbol{\mu}^{\pi_\mu})^\top \mathbf{a}_{\mathbf{v}^*}$ is due to the mismatch between $\boldsymbol{\mu}$ and $\boldsymbol{\mu}^{\pi_\mu}$, or more specifically, the mismatch between the two marginals $\mathbf{d} = \mathbf{E}^\top \boldsymbol{\mu}$ and $\mathbf{d}^{\pi_\mu} = \mathbf{E}^\top \boldsymbol{\mu}^{\pi_\mu}$. Indeed, when $\mathbf{d} = \mathbf{d}^{\pi_\mu}$, the residual is equal to the performance gap. However, in general, we do not have control over that difference for the sequence of variables $\{x_n = (\mathbf{v}_n, \boldsymbol{\mu}_n) \in \mathcal{X}\}$ an algorithm generates. The sufficient condition in Proposition 9.3.1 attempts to mitigate the difference, using the fact $\mathbf{d}^{\pi_\mu} = (1 - \gamma)\mathbf{p} + \gamma \mathbf{P}_{\pi_\mu}^\top \mathbf{d}^{\pi_\mu}$ from (9.2), where $\mathbf{P}_{\pi_\mu}$ is the transition matrix under $\pi_{\boldsymbol{\mu}}$. But the missing half $\gamma \mathbf{P}_{\pi_\mu}^\top \mathbf{d}^{\pi_\mu}$ (due to the long-term effects in the MDP) introduces the

unavoidable, weak constant $(1-\gamma)\min_s p(s)$, if we want to have an uniform bound on $\|\mathbf{v}^* - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty$. The counterexample in Proposition 9.3.2 was designed to maximize the effect of covariate shift, so that $\boldsymbol{\mu}$ fails to captures state-action pairs with high advantage. To break the curse, we must properly weight the gap between $\mathbf{v}^*$ and $\mathbf{v}^{\pi_{\boldsymbol{\mu}}}$ instead of relying on the uniform bound on $\|\mathbf{v}^* - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty$ as before.

9.4 The Reduction

The analyses above reveal both good and bad properties of the saddle-point setup in (9.8). On the one hand, we showed that approximate solutions to the saddle-point problem in (9.8) can be obtained by running any no-regret algorithm in the single-player COL problem defined in (9.16); many efficient algorithms are available from the online learning literature. On the other hand, we also discovered a root difficulty in converting an approximate solution of (9.8) to an approximately optimal policy in RL (Proposition 9.3.1), even after imposing strong conditions like (9.19). At this point, one may wonder if the formulation based on (9.8) is fundamentally sample inefficient compared with other approaches to RL, but this is actually not true.

Our main contribution shows that learning a policy through running a no-regret algorithm in the COL problem in (9.16) is, in fact, as sample efficient in policy performance as other RL techniques, even without the common constraint in (9.19) or extra assumptions on the MDP like ergodicity imposed in the literature.

Theorem 9.4.1. *Let* $X_N = \{x_n \in \mathcal{X}\}_{n=1}^N$ *be* any *sequence. Let* $\hat{\pi}_N$ *be the policy given by* $\hat{x}_N$ *via* (9.17)*, which is either the average or the best decision in* X_N*. Define* $y_N^* \coloneqq (\mathbf{v}^{\hat{\pi}_N}, \boldsymbol{\mu}^*)$*. Then* $V^{\hat{\pi}_N}(p) \geq V^*(p) - \frac{\mathrm{regret}_N(y_N^*)}{N}$.

Theorem 9.4.1 shows that if X_N has sublinear regret, then both the average policy and the best policy in X_N converge to the optimal policy in performance with a rate $O(\mathrm{regret}_N(y_N^*)/N)$. Compared with existing results obtained through Proposition 9.3.1,

the above result removes the factor $(1-\gamma)\min_s p(s)$ and does not impose any assumption on X_N or the MDP. Indeed Theorem 9.4.1 holds for *any* sequence. For example, when X_N is generated by stochastic feedback of l_n, Theorem 9.4.1 continues to hold, as the regret is defined in terms of l_n, not of the sampled loss. Stochasticity will only affect the rate of regret.

In other words, we have shown that when $\boldsymbol{\mu}$ and $\mathbf{v}$ can be directly parameterized, an approximately optimal policy for the RL problem can be obtained by running any no-regret online learning algorithm, and that the policy quality is simply dictated by the regret rate. To illustrate, in Section 9.5 we will prove that simply running mirror descent in this COL produces an RL algorithm that is as sample efficient as other common RL techniques. One can further foresee that algorithms leveraging the continuity in COL—e.g. mirror-prox (Juditsky, Nemirovski, and Tauvel, 2011) or PicCoLO (Cheng et al., 2019d)—and variance reduction can lead to more sample efficient RL algorithms.

Below we will also demonstrate how to use the fact that COL is single-player (see Section 9.2.3) to cleanly incorporate the effects of using function approximators to model $\boldsymbol{\mu}$ and $\mathbf{v}$. We will present a corollary of Theorem 9.4.1, which separates the problem of *learning* $\boldsymbol{\mu}$ and $\mathbf{v}$, and that of *approximating* $\mathcal{M}$ and $\mathcal{V}$ with function approximators. The first part is controlled by the rate of regret in online learning, and the second part depends on only the chosen class of function approximators, independently of the learning process. As these properties are agnostic to problem setups and algorithms, our reduction leads to a framework for systematic synthesis of new RL algorithms with performance guarantees. The missing proofs of this section are in Section 9.C.

9.4.1 Proof of Theorem 9.4.1

The main insight of our reduction is to adopt, in defining $r_{ep}(x;x')$, a comparator $x' \in \mathcal{X}$ based on the output of the algorithm (represented by x), instead of the fixed comparator x^* (the optimal pair of value function and state-action distribution) that has been used

conventionally, e.g. in Proposition 9.3.1. While this idea seems unnatural from the standard saddle-point or EP perspective, it is possible, because the regret in online learning is measured against the worst-case choice in $\mathcal{X}$, which is allowed to be selected in *hindsight*. Specifically, we propose to select the following comparator to directly bound $V^*(p) - V^{\hat{\pi}_N}(p)$ instead of the conservative measure $\|V^* - V^{\hat{\pi}_N}\|_\infty$ used before.

Proposition 9.4.1. *For $x = (\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{X}$, define $y_x^* \coloneqq (\mathbf{v}^{\pi_{\boldsymbol{\mu}}}, \boldsymbol{\mu}^*) \in \mathcal{X}$. It holds $r_{ep}(x; y_x^*) = V^*(p) - V^{\pi_{\boldsymbol{\mu}}}(p)$.*

To finish the proof, let $\hat{x}_N$ be either $\frac{1}{N}\sum_{n=1}^N x_n$ or $\arg\min_{x \in X_N} r_{ep}(x; y_x^*)$, and let $\hat{\pi}_N$ denote the policy given by (9.17). First, $V^*(p) - V^{\hat{\pi}_N}(p) = r_{ep}(\hat{x}_N; y_N^*)$ by Proposition 9.4.1. Next we follow the proof of Proposition 9.2.1 (Cheng et al., 2019c): because F is skew-symmetric and $F(y_N^*, \cdot)$ is convex, we have by (9.18)

$$\begin{aligned} &V^*(p) - V^{\hat{\pi}_N}(p) = r_{ep}(\hat{x}_N; y_N^*) = -F(\hat{x}_N, y_N^*) \\ &= F(y_N^*, \hat{x}_N) \leq \tfrac{1}{N}\textstyle\sum_{n=1}^N F(y_N^*, x_n) \\ &= \tfrac{1}{N}\textstyle\sum_{n=1}^N -F(x_n, y_N^*) = \tfrac{1}{N}\mathrm{regret}_N(y_N^*) \end{aligned}$$

9.4.2 Function Approximators

When the state and action spaces are large or continuous, directly optimizing $\mathbf{v}$ and $\boldsymbol{\mu}$ can be impractical. Instead we can consider optimizing over a subset of feasible choices parameterized by function approximators

$$\mathcal{X}_\Theta = \{\mathbf{x}_\theta = (\boldsymbol{\phi}_\theta, \boldsymbol{\psi}_\theta) : \boldsymbol{\psi}_\theta \in \mathcal{M}, \theta \in \Theta\}, \tag{9.20}$$

where $\boldsymbol{\phi}_\theta$ and $\boldsymbol{\psi}_\theta$ are functions parameterized by $\theta \in \Theta$, and Θ is a parameter set. Because COL is a single-player setup, we can extend the previous idea and Theorem 9.4.1 to provide performance bounds in this case by a simple rearrangement (see Section 9.C), which is a common trick used in the online imitation learning literature (Cheng and Boots, 2018;

Ross, Gordon, and Bagnell, 2011). Notice that, in (9.20), we require only $\boldsymbol{\psi}_\theta \in \mathcal{M}$, but not $\boldsymbol{\phi}_\theta \in \mathcal{V}$, because for the performance bound in our reduction to hold, we only need the constraint $\mathcal{M}$ (see Lemma 9.C.2 in proof of Proposition 9.4.1).

Corollary 9.4.1. *Let $X_N = \{x_n \in \mathcal{X}_\theta\}_{n=1}^N$ be any sequence. Let $\hat{\pi}_N$ be the policy given either by the average or the best decision in X_N. It holds that*

$$V^{\hat{\pi}_N}(p) \geq V^*(p) - \frac{\mathrm{regret}_N(\Theta)}{N} - \epsilon_{\Theta,N}$$

where $\epsilon_{\Theta,N} = \min_{x_\theta \in \mathcal{X}_\theta} r_{ep}(\hat{x}_N; y_N^*) - r_{ep}(\hat{x}_N; x_\theta)$ measures the expressiveness of X_θ, and $\mathrm{regret}_N(\Theta) \coloneqq \sum_{n=1}^N l_n(x_n) - \min_{x \in \mathcal{X}_\Theta} \sum_{n=1}^N l_n(x)$.

We can quantify $\epsilon_{\Theta,N}$ with the basic Hölder's inequality.

Proposition 9.4.2. *Regardless of the parameterization, under the setup in Corollary 9.4.1, let $\hat{x}_N = (\hat{\mathbf{v}}_N, \hat{\boldsymbol{\mu}}_N)$. Then $\epsilon_{\Theta,N}$ is no larger than*

$$\begin{aligned}
&\min_{(\mathbf{v}_\theta, \boldsymbol{\mu}_\theta) \in \mathcal{X}_\Theta} \frac{\|\boldsymbol{\mu}_\theta - \boldsymbol{\mu}^*\|_1}{1-\gamma} + \min_{\mathbf{w}:\mathbf{w}\geq 1} \|\mathbf{b}_{\hat{\boldsymbol{\mu}}_N}\|_{1,\mathbf{w}} \|\mathbf{v}_\theta - \mathbf{v}^{\hat{\pi}_N}\|_{\infty,1/\mathbf{w}} \\
&\leq \min_{(\mathbf{v}_\theta, \boldsymbol{\mu}_\theta) \in \mathcal{X}_\Theta} \frac{1}{1-\gamma} \left(\|\boldsymbol{\mu}_\theta - \boldsymbol{\mu}^*\|_1 + 2\|\mathbf{v}_\theta - \mathbf{v}^{\hat{\pi}_N}\|_\infty \right).
\end{aligned}$$

where the norms are defined as $\|\mathbf{x}\|_{1,\mathbf{w}} = \sum_i w_i |x_i|$ and $\|\mathbf{x}\|_{\infty,1/\mathbf{w}} = \max_i w_i^{-1} |x_i|$.

Proposition 9.4.2 says $\epsilon_{\Theta,N}$ depends on how well $\mathcal{X}_\Theta$ captures the value function of the output policy $\mathbf{v}^{\hat{\pi}_N}$ and the optimal state-action distribution $\boldsymbol{\mu}^*$. We remark that this result is independent of how $\mathbf{v}^{\hat{\pi}_N}$ is generated. Furthermore, Proposition 9.4.2 makes *no* assumption on the structure of function approximators. It even allows sharing parameters θ between $\mathbf{v} = \boldsymbol{\phi}_\theta$ and $\boldsymbol{\mu} = \boldsymbol{\psi}_\theta$, e.g., they can be a bi-headed neural network, which is common for learning shared feature representations. More precisely, the structure of the function approximator would only affect whether $l_n((\boldsymbol{\phi}_\theta, \boldsymbol{\psi}_\theta))$ remains a convex function in θ, which determines the difficulty of designing algorithms with sublinear regret.

Algorithm 3 Mirror descent for RL

Input: ϵ optimality of the γ-average return
δ maximal failure probability
generative model of an MDP
Output: $\hat{\pi}_N = \pi^{\hat{\boldsymbol{\mu}}_N}$

1: $x_1 = (\mathbf{v}_1, \boldsymbol{\mu}_1)$ where $\boldsymbol{\mu}_1$ is uniform and $\mathbf{v}_1 \in \mathcal{V}$
2: Set $N = \tilde{\Omega}(\frac{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}{(1-\gamma)^2\epsilon^2})$ and $\eta = (1-\gamma)(|\mathcal{S}||\mathcal{A}|N)^{-1/2}$
3: Set the Bregman divergence as (9.22)
4: **for** $n = 1 \dots N-1$ **do**
5: Sample g_n according to (9.24)
6: Update to x_{n+1} according to (10.7)
7: **end for**
8: Set $(\hat{\mathbf{v}}_N, \hat{\boldsymbol{\mu}}_N) = \hat{x}_N = \frac{1}{N}\sum_{n=1}^{N} x_n$

In other words, the proposed COL formulation provides a reduction which dictates the policy performance with two separate factors: 1) the rate of regret $\mathrm{regret}_N(\Theta)$ which is controlled by the choice of online learning algorithm; 2) the approximation error $\epsilon_{\Theta,N}$ which is determined by the choice of function approximators. These two factors can almost be treated independently, except that the choice of function approximators would determine the properties of $l_n((\boldsymbol{\phi}_\theta, \boldsymbol{\psi}_\theta))$ as a function of θ, and the choice of Θ needs to ensure (9.20) is admissible.

9.5 Sample Complexity of Mirror Descent

We demonstrate the power of our reduction by applying perhaps the simplest online learning algorithm, mirror descent, to the proposed COL problem in (9.16) with stochastic feedback (Algorithm 3). For transparency, we discuss the tabular setup. We will show a natural extension to basis functions at the end.

Recall that mirror descent is a first-order algorithm, whose update rule can be written as

$$x_{n+1} = \arg\min_{x \in \mathcal{X}} \langle g_n, x \rangle + \tfrac{1}{\eta} B_R(x||x_n) \tag{9.21}$$

where $\eta > 0$ is the step size, g_n is the feedback direction, and $B_R(x||x') = R(x) - R(x') -$

$\langle \nabla R(x'), x - x' \rangle$ is the Bregman divergence generated by a strictly convex function R. Based on the geometry of $\mathcal{X} = \mathcal{V} \times \mathcal{M}$, we consider a natural Bregman divergence of the form

$$B_R(x'||x) = \frac{1}{2|\mathcal{S}|}\|\mathbf{v}' - \mathbf{v}\|_2^2 + KL(\boldsymbol{\mu}'||\boldsymbol{\mu}) \tag{9.22}$$

This choice mitigates the effects of dimension (e.g. if we set $x_1 = (\mathbf{v}_1, \boldsymbol{\mu}_1)$ with $\boldsymbol{\mu}_1$ being the uniform distribution, it holds $B_R(x'||x_1) = \tilde{O}(1)$ for any $x' \in \mathcal{X}$).

To define the feedback direction g_n, we slightly modify the per-round loss l_n in (9.16) and consider a new loss

$$h_n(x) \coloneqq \mathbf{b}_{\boldsymbol{\mu}_n}^\top \mathbf{v} + \boldsymbol{\mu}^\top (\tfrac{1}{1-\gamma}\mathbf{1} - \mathbf{a}_{\mathbf{v}_n}) \tag{9.23}$$

that shifts l_n by a constant, where $\mathbf{1}$ is the vector of ones. (The decisions generated by the mirror descent rule makes sure $\|\boldsymbol{\mu}_n\|_1 = 1$.) One can verify that $l_n(x) - l_n(x') = h_n(x) - h_n(x')$, for all x, x'. Therefore, using h_n does not change regret. The reason for using h_n instead of l_n is to make $\nabla_{\boldsymbol{\mu}} h_n((\mathbf{v}, \boldsymbol{\mu}))$ (and its unbiased approximation) a positive vector, so that the regret bound can have a better dimension dependency. This is a common trick used in online learning (e.g. EXP3) for optimizing variables living in a simplex (the $\boldsymbol{\mu}$ here).

We set the first-order feedback g_n as an unbiased *sampled* estimate of $\nabla h_n(x_n)$. In round n, this is realized by two independent calls of the generative model:

$$g_n = \begin{bmatrix} \tilde{\mathbf{p}}_n + \frac{1}{1-\gamma}(\gamma \tilde{\mathbf{P}}_n - \mathbf{E}_n)^\top \tilde{\boldsymbol{\mu}}_n \\ |\mathcal{S}||\mathcal{A}|(\frac{1}{1-\gamma}\hat{\mathbf{1}}_n - \hat{\mathbf{r}}_n - \frac{1}{1-\gamma}(\gamma \hat{\mathbf{P}}_n - \hat{\mathbf{E}}_n)\mathbf{v}_n) \end{bmatrix} \tag{9.24}$$

Let $g_n = [\mathbf{g}_{n,v}; \mathbf{g}_{n,\mu}]$. For $\mathbf{g}_{n,v}$, we sample $\mathbf{p}$, sample $\boldsymbol{\mu}_n$ to get a state-action pair, and query the transition $\mathbf{P}$ at the state-action pair sampled from $\boldsymbol{\mu}_n$. ($\tilde{\mathbf{p}}_n$, $\tilde{\mathbf{P}}_n$, and $\tilde{\boldsymbol{\mu}}_n$ denote

the single-sample estimate of these probabilities.) For $\mathbf{g}_{n,\mu}$, we first sample *uniformly* a state-action pair (which explains the factor $|\mathcal{S}||\mathcal{A}|$), and then query the reward $\mathbf{r}$ and the transition $\mathbf{P}$. ($\hat{\mathbf{1}}_n$, $\hat{\mathbf{r}}_n$, $\hat{\mathbf{P}}_n$, and $\hat{\mathbf{E}}_n$ denote the single-sample estimates.) To emphasize, we use $\tilde{\cdot}$ and $\hat{\cdot}$ to distinguish the empirical quantities obtained by these two independent queries. By construction, we have $\mathbf{g}_{n,\mu} \geq 0$. It is clear that this direction g_n is unbiased, i.e. $\mathbb{E}[g_n] = \nabla h_n(x_n)$. Moreover, it is extremely sparse and can be computed using $O(1)$ sample, computational, and memory complexities.

Below we show this algorithm, despite being extremely simple, has strong theoretical guarantees. In other words, we obtain simpler versions of the algorithms proposed in (Chen, Li, and Wang, 2018; Wang, 2017b; Wang and Chen, 2016) but with improved performance.

Theorem 9.5.1. *With probability* $1-\delta$*, Algorithm 3 learns an* ϵ*-optimal policy with* $\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}{(1-\gamma)^2\epsilon^2}\right)$ *samples.*

Note that the above statement makes no assumption on the MDP (except the tabular setup for simplifying analysis). Also, because the definition of value function in (9.1) is scaled by a factor $(1-\gamma)$, the above result translates into a sample complexity in $\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}{(1-\gamma)^4\epsilon^2}\right)$ for the conventional discounted accumulated rewards.

9.5.1 Proof Sketch of Theorem 9.5.1

The proof is based on the basic property of mirror descent and martingale concentration. We provide a sketch here; please refer to Section 9.D for details. Let $y_N^* = (\mathbf{v}^{\hat{\pi}_N}, \boldsymbol{\mu}^*)$. We bound the regret in Theorem 9.4.1 by the following rearrangement, where the equality is because h_n is a constant shift from l_n.

$$\begin{aligned}
&\mathrm{regret}_N(y_N^*) = \sum_{n=1}^N h_n(x_n) - \sum_{n=1}^N h_n(y_N^*) \\
&\leq \left(\sum_{n=1}^N (\nabla h_n(x_n) - g_n)^\top x_n\right) + \left(\max_{x\in\mathcal{X}} \sum_{n=1}^N g_n^\top (x_n - x)\right) + \left(\sum_{n=1}^N (g_n - \nabla h_n(x_n))^\top y_N^*\right)
\end{aligned}$$

We recognize the first term is a martingale, because x_n does not depend on g_n. Therefore, we can appeal to a Bernstein-type martingale concentration and prove it is in $\tilde{O}(\frac{\sqrt{N|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}}{1-\gamma})$. For the second term, by treating $g_n^\top x$ as the per-round loss, we can use standard regret analysis of mirror descent and show a bound in $\tilde{O}(\frac{\sqrt{N|\mathcal{S}||\mathcal{A}|}}{1-\gamma})$. For the third term, because $\mathbf{v}^{\hat{\pi}_N}$ in $y_N^* = (\mathbf{v}^{\hat{\pi}_N}, \boldsymbol{\mu}^*)$ depends on $\{g_n\}_{n=1}^N$, it is *not* a martingale. Nonetheless, we are able to handle it through a union bound and show it is again no more than $\tilde{O}(\frac{\sqrt{N|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}}{1-\gamma})$. Despite the union bound, it does not increase the rate because we only need to handle $\mathbf{v}^{\hat{\pi}_N}$, not $\boldsymbol{\mu}^*$ which induces a martingale. To finish the proof, we substitute this high-probability regret bound into Theorem 9.4.1 to obtain the desired claim.

9.5.2 Extension to Function Approximators

The above algorithm assumes the tabular setup for demonstration purposes. In Section 9.E, we describe a direct extension of Algorithm 3 that uses linearly parameterized function approximators of the form $x_\theta = (\boldsymbol{\Phi}\boldsymbol{\theta}_v, \boldsymbol{\Psi}\boldsymbol{\theta}_\mu)$, where columns of bases $\boldsymbol{\Phi}, \boldsymbol{\Psi}$ belong to $\mathcal{V}$ and $\mathcal{M}$, respectively, and $(\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu) \in \Theta$.

Overall the algorithm stays the same, except the gradient is computed by chain-rule, which can be done in $O(dim(\Theta))$ time and space. While this seems worse, the computational complexity per update actually improves to $O(dim(\Theta))$ from the slow $O(|\mathcal{S}||\mathcal{A}|)$ (required before for the projection in (9.22)), as now we only optimize in Θ. Moreover, we prove that its sample complexity is also better, though at the cost of bias $\epsilon_{\Theta,N}$ in Corollary 9.4.1. Therefore, the algorithm becomes applicable to large-scale or continuous problems.

Theorem 9.5.2. *Under a proper choice of Θ and B_R, with probability $1-\delta$, Algorithm 3 learns an $(\epsilon + \epsilon_{\Theta,N})$-optimal policy with $\tilde{O}\left(\frac{dim(\Theta)\log(\frac{1}{\delta})}{(1-\gamma)^2\epsilon^2}\right)$ samples.*

The proof is in Section 9.E, which follows mainly Section 9.5.1. First, we choose some Θ to satisfy (9.20) so we can use Corollary 9.4.1 to reduce the problem into regret minimization. To make the sample complexity independent of $|\mathcal{S}|$,$|\mathcal{A}|$, the key is to uniformly

sample over the columns of $\mathbf{\Psi}$, instead of over all states and actions like (9.24), in computing unbiased estimates of $\nabla_{\boldsymbol{\theta}_\mu} h_n((\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu))$. The intuition is that we should only focus on the places our basis functions care about (size of $dim(\Theta)$), instead of wasting efforts to visit all possible combinations (size of $|\mathcal{S}||\mathcal{A}|$).

9.6 Conclusion

We propose a reduction from RL to no-regret online learning that provides a systematic way to design new RL algorithms with performance guarantees. Compared with existing approaches, our framework makes no assumption on the MDP and naturally works with function approximators. To illustrate, we design a simple RL algorithm based on mirror descent; it achieves similar sample complexity as other RL techniques, but uses minimal assumptions on the MDP and is scalable to large or continuous problems. This encouraging result evidences the strength of the online learning perspective. As a future work, we believe even faster learning in RL is possible by leveraging control variate for variance reduction and by applying more advanced online techniques (Cheng et al., 2019d; Rakhlin and Sridharan, 2012) that exploit the continuity in COL to predict the future gradients.

9.A Review of RL Setups

We provide an extended review of different formulations of RL for interested readers. First, let us recall the problem setup. Let $\mathcal{S}$ and $\mathcal{A}$ be state and action spaces, and let $\pi(a|s)$ denote a policy. For $\gamma \in [0, 1)$, we are interested in solving a γ-discounted infinite-horizon RL problem:

$$\max_\pi V^\pi(p), \qquad \text{s.t.} \quad V^\pi(p) \coloneqq (1-\gamma)\mathbb{E}_{s_0\sim p}\mathbb{E}_{\xi\sim\rho_\pi(s_0)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right] \tag{9.25}$$

where $V^\pi(p)$ is the discounted average return, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function, $\rho_\pi(s_0)$ denotes the distribution of trajectory $\xi = s_0, a_0, s_1, \ldots$ generated by running π from

state s_0 in a Markov decision process (MDP), and p is a fixed but unknown initial state distribution.

9.A.1 Coordinate-wise Formulations

RL in terms of stationary state distribution Let $d_t^\pi(s)$ denote the state distribution at time t given by running π starting from p. We define its γ-weighted mixture as

$$d^\pi(s) \coloneqq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_t^\pi(s) \tag{9.26}$$

We can view d^π in (9.26) as a form of stationary state distribution of π, because it is a valid probability distribution of state and satisfies the stationarity property below,

$$d^\pi(s') = (1-\gamma)p(s') + \gamma\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}[\mathcal{P}(s'|s,a)] \tag{9.2}$$

where $\mathcal{P}(s'|s,a)$ is the transition probability of the MDP. The definition in (9.26) generalizes the concept of stationary distribution of MDP; as $\gamma \to 1$, d^π is known as the limiting average state distribution, which is the same as the stationary distribution of the MDP under π, if one exists. Moreover, with the property in (9.2), d^π summarizes the Markov structure of RL, and allows us to write (9.25) simply as

$$\max_\pi V^\pi(p), \qquad \text{s.t.} \quad V^\pi(p) = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi|s}\left[r(s,a)\right] \tag{9.27}$$

after commuting the order of expectation and summation. That is, an RL problem aims to maximize the expected reward under the stationary state-action distribution generated by the policy π.

RL in terms of value function We can also write (9.25) in terms of value function. Recall

$$V^\pi(s) \coloneqq (1-\gamma)\mathbb{E}_{\xi\sim\rho_\pi(s_0)|s_0=s}\left[\sum_{t=0}^\infty \gamma^t r(s_t,a_t)\right] \tag{9.1}$$

is the value function of π. By definition, V^π (like d^π) satisfies a stationarity property

$$V^\pi(s) = \mathbb{E}_{a\sim\pi|s}\left[(1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}|s,a}\left[V^\pi(s')\right]\right] \tag{9.5}$$

which can be viewed as a dual equivalent of (9.2). Because r is in $[0,1]$, (9.5) implies V^π lies in $[0,1]$.

The value function V^* (a shorthand of V_{π^*}) of the optimal policy π^* of the RL problem satisfies the so-called Bellman equation (Bellman, 1954): $V^*(s) = \max_{a\in\mathcal{A}}(1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}|s,a}\left[V^*(s')\right]$, where the optimal policy π^* can be recovered as the $\arg\max$. Equivalently, by the definition of $\max$, the Bellman equation amounts to finding the smallest V such that $V(s) \geq (1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}|s,a}\left[V(s')\right]$, $\forall s\in\mathcal{S}, a\in\mathcal{A}$. In other words, the RL problem in (9.25) can be written as

$$\min_V \mathbb{E}_{s\sim p}[V(s)] \qquad \text{s.t.} \quad V(s) \geq (1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}|s,a}\left[V(s')\right], \qquad \forall s\in\mathcal{S}, a\in\mathcal{A} \tag{9.28}$$

9.A.2 Linear Programming Formulations

We now connect the above two alternate expressions through the classical LP setup of RL Denardo and Fox, 1968; Manne, 1959.

LP in terms of value function The classic LP formulation[5] is simply a restatement of (9.28):

$$\min_{\mathbf{v}} \quad \mathbf{p}^\top \mathbf{v} \qquad \text{s.t.} \quad (1-\gamma)\mathbf{r} + \gamma \mathbf{P}\mathbf{v} \leq \mathbf{E}\mathbf{v} \tag{9.4}$$

where $\mathbf{p} \in \mathbb{R}^{|\mathcal{S}|}$, $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$, and $\mathbf{r} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ are the vector forms of p, V, r, respectively, $\mathbf{P} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|}$ is the transition probability[6], and $\mathbf{E} = \mathbf{I} \otimes \mathbf{1} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|}$ (we use $|\cdot|$ to denote the cardinality of a set, $\otimes$ the Kronecker product, $\mathbf{I} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ is the identity, and $\mathbf{1} \in \mathbb{R}^{|\mathcal{A}|}$ a vector of ones). It is easy to verify that for all $\mathbf{p} > 0$, the solution to (9.4) is the same and equal to $\mathbf{v}^*$ (the vector form of V^*).

LP in terms of stationary state-action distribution Define the Lagrangian function

$$\mathcal{L}(\mathbf{v}, \mathbf{f}) \coloneqq \mathbf{p}^\top \mathbf{v} + \mathbf{f}^\top ((1-\gamma)\mathbf{r} + \gamma \mathbf{P}\mathbf{v} - \mathbf{E}\mathbf{v}) \tag{9.29}$$

where $\mathbf{f} \geq \mathbf{0} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the Lagrangian multiplier. By Lagrangian duality, the dual problem of (9.4) is given as $\max_{\mathbf{f} \geq \mathbf{0}} \min_{\mathbf{v}} \mathcal{L}(\mathbf{v}, \mathbf{f})$. Or after substituting the optimality condition of $\mathbf{v}$ and define $\boldsymbol{\mu} \coloneqq (1-\gamma)\mathbf{f}$, we can write the dual problem as another LP problem

$$\max_{\boldsymbol{\mu} \geq \mathbf{0}} \quad \mathbf{r}^\top \boldsymbol{\mu} \qquad \text{s.t.} \quad (1-\gamma)\mathbf{p} + \gamma \mathbf{P}^\top \boldsymbol{\mu} = \mathbf{E}^\top \boldsymbol{\mu} \tag{9.3}$$

Note that this problem like (9.4) is normalized: we have $\|\boldsymbol{\mu}\|_1 = 1$ because $\|\mathbf{p}\|_1 = 1$, and

$$\|\boldsymbol{\mu}\|_1 = \mathbf{1}^\top \mathbf{E}^\top \boldsymbol{\mu} = (1-\gamma)\mathbf{1}^\top \mathbf{p} + \gamma \mathbf{1}^\top \mathbf{P}^\top \boldsymbol{\mu} = (1-\gamma)\|\mathbf{p}\|_1 + \gamma \|\boldsymbol{\mu}\|_1$$

[5] Our setup in (9.4) differs from the classic one in the $(1-\gamma)$ factor in the constraint to normalize the problem.

[6] We arrange the coordinates in a way such that along the $|\mathcal{S}||\mathcal{A}|$ indices are contiguous in actions.

where we use the facts that $\boldsymbol{\mu} \geq \mathbf{0}$ and $\mathbf{P}$ is a stochastic transition matrix. This means that $\boldsymbol{\mu}$ is a valid state-action distribution, from which we see that the equality constraint in (9.3) is simply a vector form (9.2). Therefore, (9.3) is the same as (9.27) if we define the policy π as the conditional distribution based on $\boldsymbol{\mu}$.

9.B Missing Proofs of Section 9.3

9.B.1 Proof of Lemma 9.3.1

Lemma 9.3.1. *For any $x = (\mathbf{v}, \boldsymbol{\mu})$, if $x' \in \mathcal{X}$ satisfies* (9.2) *and* (9.5) *(i.e. $\mathbf{v}'$ and $\boldsymbol{\mu}'$ are the value function and state-action distribution of policy $\pi_{\boldsymbol{\mu}'}$), $r_{ep}(x; x') = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'}$.*

Proof. First note that $F(x, x) = 0$. Then as x' satisfies stationarity, we can use Lemma 9.3.2 below and write

$$\begin{aligned}
r_{ep}(x; x') &= F(x, x) - F(x, x') \\
&= -F(x, x') \\
&= -(\mathbf{p}^\top \mathbf{v}' - \mathbf{p}^\top \mathbf{v}) - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} + \boldsymbol{\mu}'^\top \mathbf{a}_{\mathbf{v}} && (\because \text{Definition of } F \text{ in (9.15)}) \\
&= -\boldsymbol{\mu}' \mathbf{a}_{\mathbf{v}} - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} + \boldsymbol{\mu}'^\top \mathbf{a}_{\mathbf{v}} && (\because \text{Lemma 9.3.2}) \\
&= -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'}
\end{aligned}$$

■

9.B.2 Proof of Lemma 9.3.2

Lemma 9.3.2. *Let $\mathbf{v}^\pi$ and $\boldsymbol{\mu}^\pi$ denote the value and state-action distribution of some policy π. Then for* any *function $\mathbf{v}'$, it holds that $\mathbf{p}^\top(\mathbf{v}^\pi - \mathbf{v}') = (\boldsymbol{\mu}^\pi)^\top \mathbf{a}_{\mathbf{v}'}$. In particular, it implies $V^\pi(p) - V^{\pi'}(p) = (\boldsymbol{\mu}^\pi)^\top \mathbf{a}_{\mathbf{v}^{\pi'}}$.*

Proof. This is the well-known performance difference lemma. The proof is based on the

stationary properties in (9.2) and (9.5), which can be stated in vector form as

$$(\boldsymbol{\mu}^\pi)^\top \mathbf{E}\mathbf{v}^\pi = (\boldsymbol{\mu}^\pi)^\top((1-\gamma)\mathbf{r} + \gamma\mathbf{P}\mathbf{v}^\pi) \qquad \text{and} \qquad (1-\gamma)\mathbf{p} + \gamma\mathbf{P}^\top\boldsymbol{\mu}^\pi = \mathbf{E}^\top\boldsymbol{\mu}^\pi$$

The proof is a simple application of these two properties.

$$\begin{aligned}
\mathbf{p}^\top(\mathbf{v}^\pi - \mathbf{v}') &= \frac{1}{1-\gamma}(\mathbf{E}^\top\boldsymbol{\mu}^\pi - \gamma\mathbf{P}^\top\boldsymbol{\mu}^\pi)^\top(\mathbf{v}^\pi - \mathbf{v}') \\
&= \frac{1}{1-\gamma}(\boldsymbol{\mu}^\pi)^\top((\mathbf{E}-\gamma\mathbf{P})\mathbf{v}^\pi - (\mathbf{E}-\gamma\mathbf{P})\mathbf{v}') \\
&= \frac{1}{1-\gamma}(\boldsymbol{\mu}^\pi)^\top((1-\gamma)\mathbf{r} - (\mathbf{E}-\gamma\mathbf{P})\mathbf{v}') = (\boldsymbol{\mu}^\pi)^\top\mathbf{a}_{\mathbf{v}'}
\end{aligned}$$

where we use the stationarity property of $\boldsymbol{\mu}^\pi$ in the first equality and that $\mathbf{v}^\pi$ in the third equality. ■

9.B.3 Proof of Proposition 9.3.1

Proposition 9.3.1. *For any* $x = (\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{X}$, *if* $\mathbf{E}^\top\boldsymbol{\mu} \geq (1-\gamma)\mathbf{p}$, $r_{ep}(x; x^*) \geq (1-\gamma)\min_s p(s)\|\mathbf{v}^* - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty$.

Proof. This proof mainly follows the steps in Wang, 2017b but written in our notation. First Lemma 9.3.1 shows $r_{ep}(x; x^*) = -\boldsymbol{\mu}^\top\mathbf{a}_{\mathbf{v}^*}$. We then lower bound $-\boldsymbol{\mu}^\top\mathbf{a}_{\mathbf{v}^*}$ by reversing the proof of the performance difference lemma (Lemma 9.3.2).

$$\begin{aligned}
\boldsymbol{\mu}^\top\mathbf{a}_{\mathbf{v}^*} &= \frac{1}{1-\gamma}\boldsymbol{\mu}^\top((1-\gamma)\mathbf{r} - (\mathbf{E}-\gamma\mathbf{P})\mathbf{v}^*) && (\because \text{Definition of } \mathbf{a}_{\mathbf{v}^*}) \\
&= \frac{1}{1-\gamma}\boldsymbol{\mu}^\top((\mathbf{E}-\gamma\mathbf{P})\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - (\mathbf{E}-\gamma\mathbf{P})\mathbf{v}^*) && (\because \text{Stationarity of } \mathbf{v}^{\pi_{\boldsymbol{\mu}}}) \\
&= \frac{1}{1-\gamma}\boldsymbol{\mu}^\top(\mathbf{E}-\gamma\mathbf{P})(\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - \mathbf{v}^*) \\
&= \frac{1}{1-\gamma}\mathbf{d}^\top(\mathbf{I}-\gamma\mathbf{P}_{\pi_{\boldsymbol{\mu}}})(\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - \mathbf{v}^*)
\end{aligned}$$

where we define $\mathbf{d} \coloneqq \mathbf{E}^\top\boldsymbol{\mu}$ and $\mathbf{P}_{\pi_{\boldsymbol{\mu}}}$ as the state-transition of running policy $\pi_{\boldsymbol{\mu}}$.

We wish to further upper bound this quantity. To proceed, we appeal to the Bellman equation of the optimal value function $\mathbf{v}^*$ and the stationarity of $\mathbf{v}^{\pi_{\boldsymbol{\mu}}}$:

$$\mathbf{v}^* \geq (1-\gamma)\mathbf{r}_{\pi_{\boldsymbol{\mu}}} + \gamma \mathbf{P}_{\pi_{\boldsymbol{\mu}}}\mathbf{v}^* \qquad \text{and} \qquad \mathbf{v}^{\pi_{\boldsymbol{\mu}}} = (1-\gamma)\mathbf{r}_{\pi_{\boldsymbol{\mu}}} + \gamma \mathbf{P}_{\pi_{\boldsymbol{\mu}}}\mathbf{v}^{\pi_{\boldsymbol{\mu}}},$$

which together imply that $(\mathbf{I} - \gamma \mathbf{P}_{\pi_{\boldsymbol{\mu}}})(\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - \mathbf{v}^*) \leq 0$. We will also use the stationarity of $\mathbf{d}^{\pi_{\boldsymbol{\mu}}}$ (the average state distribution of $\pi_{\boldsymbol{\mu}}$): $\mathbf{d}^{\pi_{\boldsymbol{\mu}}} = (1-\gamma)\mathbf{p} + \gamma \mathbf{P}_{\boldsymbol{\pi}_{\boldsymbol{\mu}}}^\top \mathbf{d}^{\pi_{\boldsymbol{\mu}}}$.

Since $\mathbf{d} \geq (1-\gamma)\mathbf{p}$ in the assumption, we can then write

$$\begin{aligned}
\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*} &= \frac{1}{1-\gamma}\mathbf{d}^\top (\mathbf{I} - \gamma \mathbf{P}_{\pi_{\boldsymbol{\mu}}})(\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - \mathbf{v}^*) \\
&\leq \mathbf{p}^\top (\mathbf{I} - \gamma \mathbf{P}_{\pi_{\boldsymbol{\mu}}})(\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - \mathbf{v}^*) \\
&\leq -\min_s p(s) \|(\mathbf{I} - \gamma \mathbf{P}_{\pi_{\boldsymbol{\mu}}})(\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - \mathbf{v}^*)\|_\infty \\
&\leq -\min_s p(s)(1-\gamma)\|\mathbf{v}^{\pi_{\boldsymbol{\mu}}} - \mathbf{v}^*\|_\infty.
\end{aligned}$$

Finally, flipping the sign of the inequality concludes the proof. ■

9.B.4 Proof of Proposition 9.3.2

Proposition 9.3.2. *There is a class of MDPs such that, for some $x \in \mathcal{X}$, Proposition 9.3.1 is an equality.*

Proof. We show this equality holds for a class of MDPs. For simplicity, let us first consider an MDP with three states $1, 2, 3$ and for each state there are three actions ($left$, $right$, $stay$). They correspond to an intuitive, deterministic transition dynamics

$$\mathcal{P}(\max\{s-1, 1\}|s, left) = 1, \quad \mathcal{P}(\min\{s+1, 3\}|s, right) = 1, \quad \mathcal{P}(s|s, stay) = 1.$$

We set the reward as $r(s, right) = 1$ for $s = 1, 2, 3$ and zero otherwise. It is easy to see that the optimal policy is $\pi^*(right|s) = 1$, which has value function $\mathbf{v}^* = [1, 1, 1]^\top$.

Now consider $x = (\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{X}$. To define $\boldsymbol{\mu}$, let $\mu(s, a) = d(s)\pi_{\boldsymbol{\mu}}(a|s)$. We set

$$\pi_{\boldsymbol{\mu}}(right|1) = 1, \quad \pi_{\boldsymbol{\mu}}(stay|2) = 1, \qquad \pi_{\boldsymbol{\mu}}(right|3) = 1$$

That is, $\pi_{\boldsymbol{\mu}}$ is equal to π^* except when $s = 2$. One can verify the value function of this policy is $\mathbf{v}^{\pi_\mu} = [(1-\gamma), 0, 1]^\top$.

As far as d is concerned ($\mathbf{d} = \mathbf{E}^\top \boldsymbol{\mu}$), suppose the initial distribution is uniform, i.e. $\mathbf{p} = [1/3, 1/3, 1/3]^\top$, we choose d as $\mathbf{d} = (1-\gamma)\mathbf{p} + \gamma[1, 0, 0]^\top$, which satisfies the assumption in Proposition 9.3.1. Therefore, we have $\boldsymbol{\mu} \in \mathcal{M}'$ and we will let $\mathbf{v}$ be some arbitrary point in $\mathcal{V}$.

Now we show for this choice $x = (\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{V} \times \mathcal{M}'$, the equality in Proposition 9.3.1 holds. By Lemma 9.3.1, we know $r_{ep}(x; x') = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*}$. Recall the advantage is defined as: $\mathbf{a}_{\mathbf{v}^*} = \mathbf{r} + \frac{1}{1-\gamma}(\gamma\mathbf{P} - \mathbf{E})\mathbf{v}^*$. Let $A_{V^*}(s, a)$ denote the functional form of $\mathbf{a}_{\mathbf{v}^*}$ and define the expected advantage:

$$A_{V^*}(s, \pi_{\boldsymbol{\mu}}) \coloneqq \mathbb{E}_{a\sim\pi_{\boldsymbol{\mu}}}[A_{V^*}(s, a)].$$

We can verify it has the following values:

$$A_{V^*}(1, \pi_{\boldsymbol{\mu}}) = 0, \quad A_{V^*}(2, \pi_{\boldsymbol{\mu}}) = -1, \quad A_{V^*}(3, \pi_{\boldsymbol{\mu}}) = 0.$$

Thus, the above construction yields

$$r_{ep}(x; x^*) = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*} = \frac{(1-\gamma)}{3} = (1-\gamma)\min_s p(s)\|\mathbf{v}^* - \mathbf{v}^{\pi_\mu}\|_\infty$$

One can easily generalize this 3-state MDP to an $|\mathcal{S}|$-state MDP where states are partitioned into three groups. ■

9.C Missing Proofs of Section 9.4

9.C.1 Proof of Proposition 9.4.1

Proposition 9.4.1. *For $x = (\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{X}$, define $y^*_x \coloneqq (\mathbf{v}^{\pi_\mu}, \boldsymbol{\mu}^*) \in \mathcal{X}$. It holds $r_{ep}(x; y^*_x) = V^*(p) - V^{\pi_\mu}(p)$.*

Proof. First we generalize Lemma 9.3.1.

Lemma 9.C.1. *Let $x = (\mathbf{v}, \boldsymbol{\mu})$ be arbitrary. Consider $\tilde{x}' = (\mathbf{v}' + \mathbf{u}', \boldsymbol{\mu}')$, where $\mathbf{v}'$ and $\boldsymbol{\mu}'$ are the value function and state-action distribution of policy $\pi_{\boldsymbol{\mu}'}$, and $\mathbf{u}'$ is arbitrary. It holds that $r_{ep}(x; \tilde{x}') = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} - \mathbf{b}_{\boldsymbol{\mu}}^\top \mathbf{u}'$.*

To proceed, we write $y^*_x = (\mathbf{v}^* + (\mathbf{v}^{\pi_\mu} - \mathbf{v}^*), \boldsymbol{\mu}^*)$ and use Lemma 9.C.1, which gives $r_{ep}(x; y^*_x) = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*} - \mathbf{b}_{\boldsymbol{\mu}}^\top (\mathbf{v}^{\pi_\mu} - \mathbf{v}^*)$. To relate this equality to the policy performance gap, we also need the following equality.

Lemma 9.C.2. *For $\boldsymbol{\mu} \in \mathcal{M}$, it holds that $-\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*} = V^*(p) - V^{\pi_\mu}(p) + \mathbf{b}_{\boldsymbol{\mu}}^\top (\mathbf{v}^{\pi_\mu} - \mathbf{v}^*)$.*

Together they imply the desired equality $r_{ep}(x; y^*_x) = V^*(p) - V^{\pi_\mu}(p)$. ■

Proof of Lemma 9.C.1

Lemma 9.C.1. *Let $x = (\mathbf{v}, \boldsymbol{\mu})$ be arbitrary. Consider $\tilde{x}' = (\mathbf{v}' + \mathbf{u}', \boldsymbol{\mu}')$, where $\mathbf{v}'$ and $\boldsymbol{\mu}'$ are the value function and state-action distribution of policy $\pi_{\boldsymbol{\mu}'}$, and $\mathbf{u}'$ is arbitrary. It holds that $r_{ep}(x; \tilde{x}') = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} - \mathbf{b}_{\boldsymbol{\mu}}^\top \mathbf{u}'$.*

Proof. Let $x' = (\mathbf{v}', \boldsymbol{\mu}')$. As shorthand, define $\mathbf{f}' \coloneqq \mathbf{v}' + \mathbf{u}'$, and $\mathbf{L} \coloneqq \frac{1}{1-\gamma}(\gamma \mathbf{P} - \mathbf{E})$ (i.e. we can write $\mathbf{a}_{\mathbf{f}} = \mathbf{r} + \mathbf{L}\mathbf{f}$). Because $r_{ep}(x; x') = -F(x, x') = -(\mathbf{p}^\top \mathbf{v}' + \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} - \mathbf{p}^\top \mathbf{v} - \boldsymbol{\mu}'^\top \mathbf{a}_{\mathbf{v}})$, we can write

$$
\begin{aligned}
r_{ep}(x; \tilde{x}') &= -\mathbf{p}^\top \mathbf{f}' - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{f}'} + \mathbf{p}^\top \mathbf{v} + \boldsymbol{\mu}'^\top \mathbf{a}_{\mathbf{v}} \\
&= \left(-\mathbf{p}^\top \mathbf{v}' - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} + \mathbf{p}^\top \mathbf{v} + \boldsymbol{\mu}'^\top \mathbf{a}_{\mathbf{v}}\right) - \mathbf{p}^\top \mathbf{u}' - \boldsymbol{\mu}^\top \mathbf{L} \mathbf{u}'
\end{aligned}
$$

$$= r_{ep}(x; x') - \mathbf{p}^\top \mathbf{u}' - \boldsymbol{\mu}^\top \mathbf{L}\mathbf{u}'$$
$$= r_{ep}(x; x') - \mathbf{b}_{\boldsymbol{\mu}}^\top \mathbf{u}'$$

Finally, by Lemma 9.3.1, we have also $r_{ep}(x; x') = -\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'}$ and therefore the final equality.

■

Proof of Lemma 9.C.2

Lemma 9.C.2. *For $\boldsymbol{\mu} \in \mathcal{M}$, it holds that $-\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^*} = V^*(p) - V^{\pi_\mu}(p) + \mathbf{b}_{\boldsymbol{\mu}}^\top (\mathbf{v}^{\pi_\mu} - \mathbf{v}^*)$.*

Proof. Following the setup in Lemma 9.C.1, we prove the statement by the rearrangement below:

$$\begin{aligned}
-\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} &= -(\boldsymbol{\mu}^{\pi_\mu})^\top \mathbf{a}_{\mathbf{v}'} + (\boldsymbol{\mu}^{\pi_\mu})^\top \mathbf{a}_{\mathbf{v}'} - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} \\
&= V^{\pi'}(p) - V^{\pi_\mu}(p) + (\boldsymbol{\mu}^{\pi_\mu} - \boldsymbol{\mu})^\top \mathbf{a}_{\mathbf{v}'} \\
&= \Big(V^{\pi'}(p) - V^{\pi_\mu}(p)\Big) + (\boldsymbol{\mu}^{\pi_\mu} - \boldsymbol{\mu})^\top \mathbf{r} + (\boldsymbol{\mu}^{\pi_\mu} - \boldsymbol{\mu})^\top \mathbf{L}\mathbf{v}'
\end{aligned}$$

where the first equality is due to the performance difference lemma, i.e. Lemma 9.3.2, and the last equality uses the definition $\mathbf{a}_{\mathbf{v}'} = \mathbf{r} + \mathbf{L}\mathbf{v}'$. For the second term, because $\boldsymbol{\mu} \in \mathcal{M}$, we can rewrite it as

$$\begin{aligned}
(\boldsymbol{\mu}^{\pi_\mu} - \boldsymbol{\mu})^\top \mathbf{r} &= (\mathbf{E}^\top \boldsymbol{\mu}^{\pi_\mu} - \mathbf{E}^\top \boldsymbol{\mu}) \mathbf{r}_{\pi_\mu} \\
&= ((1-\gamma)\mathbf{p} + \gamma \mathbf{P}^\top \boldsymbol{\mu}^{\pi_\mu} - \mathbf{E}^\top \boldsymbol{\mu}) \mathbf{r}_{\pi_\mu} \\
&= (1-\gamma)\mathbf{b}_{\boldsymbol{\mu}}^\top \mathbf{r}_{\pi_\mu} + \gamma (\boldsymbol{\mu}^{\pi_\mu} - \boldsymbol{\mu})^\top \mathbf{P}\mathbf{r}_{\pi_\mu} \\
&= (1-\gamma)\mathbf{b}_{\boldsymbol{\mu}}^\top \left(\mathbf{r}_{\pi_\mu} + \gamma \mathbf{P}_{\pi_\mu}\mathbf{r}_{\pi_\mu} + \gamma^2 \mathbf{P}_{\pi_\mu}^2 \mathbf{r}_{\pi_\mu} + \dots\right) \\
&= \mathbf{b}_{\boldsymbol{\mu}}^\top \mathbf{v}^{\pi_\mu}
\end{aligned}$$

where $\mathbf{r}_{\pi_\mu}$ and $\mathbf{P}_{\pi_\mu}$ denote the expected reward and transition under π_μ, and the second

equality uses the stationarity of $\boldsymbol{\mu}^{\pi_\mu}$ given by (9.2). For the third term, it can be written

$$(\boldsymbol{\mu}^{\pi_\mu} - \boldsymbol{\mu})^\top \mathbf{L}\mathbf{v}' = (-\mathbf{p} - \mathbf{L}^\top \boldsymbol{\mu})^\top \mathbf{v}' = -\mathbf{b}_{\boldsymbol{\mu}}^\top \mathbf{v}'$$

where the first equality uses stationarity, i.e. $\mathbf{b}_{\boldsymbol{\mu}^{\pi_\mu}} = \mathbf{p} + \mathbf{L}^\top \boldsymbol{\mu}^{\pi_\mu} = 0$. Finally combining the three steps, we have

$$-\boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}'} = V^{\pi'}(p) - V^{\pi_\mu}(p) + \mathbf{b}_{\boldsymbol{\mu}}(\mathbf{v}^{\pi_\mu} - \mathbf{v}')$$

■

9.C.2 Proof of Corollary 9.4.1

Corollary 9.4.1. *Let $X_N = \{x_n \in \mathcal{X}_\theta\}_{n=1}^N$ be any sequence. Let $\hat{\pi}_N$ be the policy given either by the average or the best decision in X_N. It holds that*

$$V^{\hat{\pi}_N}(p) \geq V^*(p) - \frac{\text{regret}_N(\Theta)}{N} - \epsilon_{\Theta,N}$$

where $\epsilon_{\Theta,N} = \min_{x_\theta \in \mathcal{X}_\theta} r_{ep}(\hat{x}_N; y_N^*) - r_{ep}(\hat{x}_N; x_\theta)$ measures the expressiveness of X_θ, and $\text{regret}_N(\Theta) \coloneqq \sum_{n=1}^N l_n(x_n) - \min_{x \in \mathcal{X}_\Theta} \sum_{n=1}^N l_n(x)$.

Proof. This can be proved by a simple rearrangement

$$\begin{aligned} V^*(p) - V^{\hat{\pi}_N}(p) &= r_{ep}(\hat{x}_N; y_N^*) \\ &= \epsilon_{\Theta,N} + \max_{x_\theta \in \mathcal{X}_\theta} r_{ep}(\hat{x}_N; x_\theta) \leq \epsilon_{\Theta,N} + \frac{\text{regret}_N(\Theta)}{N} \end{aligned}$$

where the first equality is Proposition 9.4.1 and the last inequality is due to skew-symmetry of F, similar to the proof of Theorem 9.4.1. ■

9.C.3 Proof of Proposition 9.4.2

Proposition 9.4.2. *Regardless of the parameterization, under the setup in Corollary 9.4.1, let $\hat{x}_N = (\hat{\mathbf{v}}_N, \hat{\boldsymbol{\mu}}_N)$. Then $\epsilon_{\Theta,N}$ is no larger than*

$$\begin{aligned}&\min_{(\mathbf{v}_\theta,\boldsymbol{\mu}_\theta)\in\mathcal{X}_\Theta} \frac{\|\boldsymbol{\mu}_\theta - \boldsymbol{\mu}^*\|_1}{1-\gamma} + \min_{\mathbf{w}:\mathbf{w}\geq 1} \|\mathbf{b}_{\hat{\boldsymbol{\mu}}_N}\|_{1,\mathbf{w}} \|\mathbf{v}_\theta - \mathbf{v}^{\hat{\pi}_N}\|_{\infty,1/\mathbf{w}} \\ &\leq \min_{(\mathbf{v}_\theta,\boldsymbol{\mu}_\theta)\in\mathcal{X}_\Theta} \frac{1}{1-\gamma} \left(\|\boldsymbol{\mu}_\theta - \boldsymbol{\mu}^*\|_1 + 2\|\mathbf{v}_\theta - \mathbf{v}^{\hat{\pi}_N}\|_\infty \right).\end{aligned}$$

where the norms are defined as $\|\mathbf{x}\|_{1,\mathbf{w}} = \sum_i w_i |x_i|$ and $\|\mathbf{x}\|_{\infty,1/\mathbf{w}} = \max_i w_i^{-1}|x_i|$.

Proof. For shorthand, let us set $x = (\mathbf{v}, \boldsymbol{\mu}) = \hat{x}_N$ and write also $\pi^{\boldsymbol{\mu}} = \hat{\pi}_N$ as the associated policy. Let $y_x^* = (\mathbf{v}^{\pi_\mu}, \boldsymbol{\mu}^*)$ and similarly let $x_\theta = (\mathbf{v}_\theta, \boldsymbol{\mu}_\theta) \in \mathcal{X}_\Theta$. With $r_{ep}(x; x') = -F(x, x')$ and (9.15), we can write

$$\begin{aligned}&r_{ep}(x; y_x^*) - r_{ep}(x; x_\theta) \\ &= \left(-\mathbf{p}^\top \mathbf{v}^{\pi_\mu} - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}^{\pi_\mu}} + \mathbf{p}^\top \mathbf{v} + \boldsymbol{\mu}^{*\top} \mathbf{a}_{\mathbf{v}}\right) - \left(-\mathbf{p}^\top \mathbf{v}_\theta - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}_\theta} + \mathbf{p}^\top \mathbf{v} + \boldsymbol{\mu}_\theta^\top \mathbf{a}_{\mathbf{v}}\right) \\ &= \mathbf{p}^\top (\mathbf{v}_\theta - \mathbf{v}^{\pi_\mu}) + (\boldsymbol{\mu}^* - \boldsymbol{\mu}_\theta)^\top \mathbf{a}_{\mathbf{v}} + \boldsymbol{\mu}^\top (\mathbf{a}_{\mathbf{v}_\theta} - \mathbf{a}_{\mathbf{v}^{\pi_\mu}}) \\ &= \mathbf{b}_{\boldsymbol{\mu}}^\top (\mathbf{v}_\theta - \mathbf{v}^{\pi_\mu}) + (\boldsymbol{\mu}^* - \boldsymbol{\mu}_\theta)^\top \mathbf{a}_{\mathbf{v}}\end{aligned}$$

Next we quantize the size of $\mathbf{a}_{\mathbf{v}}$ and $\mathbf{b}_{\boldsymbol{\mu}}$.

Lemma 9.C.3. *For $(\mathbf{v}, \boldsymbol{\mu}) \in \mathcal{X}$, $\|\mathbf{a}_{\mathbf{v}}\|_\infty \leq \frac{1}{1-\gamma}$ and $\|\mathbf{b}_{\boldsymbol{\mu}}\|_1 \leq \frac{2}{1-\gamma}$.*

Proof of Lemma 9.C.3. Let Δ denote the set of distributions

$$\begin{aligned}\|\mathbf{a}_{\mathbf{v}}\|_\infty &= \frac{1}{1-\gamma} \|(1-\gamma)\mathbf{r} + \gamma \mathbf{P}\mathbf{v} - \mathbf{E}\mathbf{v}\|_\infty \leq \frac{1}{1-\gamma} \max_{a,b\in[0,1]} |a-b| \leq \frac{1}{1-\gamma} \\ \|\mathbf{b}_{\boldsymbol{\mu}}\|_1 &= \frac{1}{1-\gamma} \|(1-\gamma)\mathbf{p} + \gamma \mathbf{P}^\top \boldsymbol{\mu} - \mathbf{E}^\top \boldsymbol{\mu}\|_1 \leq \frac{1}{1-\gamma} \max_{\mathbf{q},\mathbf{q}'\in\Delta} \|\mathbf{q} - \mathbf{q}'\|_1 \leq \frac{2}{1-\gamma}\end{aligned}$$

■

Therefore, we have preliminary upper bounds:

$$(\boldsymbol{\mu}^* - \boldsymbol{\mu}_\theta)^\top \mathbf{a}_\mathbf{v} \le \|\mathbf{a}_\mathbf{v}\|_\infty \|\boldsymbol{\mu}^* - \boldsymbol{\mu}_\theta\|_1 \le \frac{1}{1-\gamma}\|\boldsymbol{\mu}^* - \boldsymbol{\mu}_\theta\|_1$$
$$\mathbf{b}_{\boldsymbol{\mu}}^\top(\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}) \le \|\mathbf{b}_{\boldsymbol{\mu}}\|_1 \|\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty \le \frac{2}{1-\gamma}\|\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty$$

However, the second inequality above can be very conservative, especially when $\mathbf{b}_{\boldsymbol{\mu}} \approx 0$ which can be likely when it is close to the end of policy optimization. To this end, we introduce a free vector $\mathbf{w} \ge 1$. Define norms $\|\mathbf{v}\|_{\infty,1/\mathbf{w}} = \max_i \frac{|v_i|}{w_i}$ and $\|\boldsymbol{\delta}\|_{1,\mathbf{w}} = \sum_i w_i|\delta_i|$. Then we can instead have an upper bound

$$\mathbf{b}_{\boldsymbol{\mu}}^\top(\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}) \le \min_{\mathbf{w}:\mathbf{w}\ge 1} \|\mathbf{b}_{\boldsymbol{\mu}}\|_{1,\mathbf{w}} \|\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_{\infty,1/\mathbf{w}}$$

Notice that when $\mathbf{w} = \mathbf{1}$ the above inequality reduces to $\mathbf{b}_{\boldsymbol{\mu}}^\top(\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}) \le \|\mathbf{b}_{\boldsymbol{\mu}}\|_1 \|\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty$, which as we showed has an upper bound $\frac{2}{1-\gamma}\|\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty$.

Combining the above upper bounds, we have an upper bound on $\epsilon_{\Theta,N}$:

$$\begin{aligned}\epsilon_{\Theta,N} = r_{ep}(x; y_x^*) - r_{ep}(x; x_\theta) &\le \frac{1}{1-\gamma}\|\boldsymbol{\mu}_\theta - \boldsymbol{\mu}^*\|_1 + \min_{\mathbf{w}:\mathbf{w}\ge 1} \|\mathbf{b}_{\boldsymbol{\mu}}\|_{1,\mathbf{w}} \|\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_{\infty,1/\mathbf{w}} \\ &\le \frac{1}{1-\gamma}\left(\|\boldsymbol{\mu}_\theta - \boldsymbol{\mu}^*\|_1 + 2\|\mathbf{v}_\theta - \mathbf{v}^{\pi_{\boldsymbol{\mu}}}\|_\infty\right).\end{aligned}$$

Since it holds for any $\theta \in \Theta$, we can minimize the right-hand side over all possible choices.

■

9.D Proof of Sample Complexity of Mirror Descent

Theorem 9.5.1. *With probability $1-\delta$, Algorithm 3 learns an ϵ-optimal policy with $\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}{(1-\gamma)^2\epsilon^2}\right)$ samples.*

The proof is a combination of the basic property of mirror descent (Lemma 10.F.1) and the martingale concentration. Define $K = |\mathcal{S}||\mathcal{A}|$ and $\kappa = \frac{1}{1-\gamma}$ as shorthand. We

first slightly modify the per-round loss used to compute the gradient. Recall $l_n(x) \coloneqq \mathbf{p}^\top \mathbf{v} + \boldsymbol{\mu}_n^\top \mathbf{a}_{\mathbf{v}} - \mathbf{p}^\top \mathbf{v}_n - \boldsymbol{\mu}^\top \mathbf{a}_{\mathbf{v}_n}$ and let us consider instead a loss function

$$h_n(x) \coloneqq \mathbf{b}_{\boldsymbol{\mu}_n}^\top \mathbf{v} + \boldsymbol{\mu}^\top (\kappa \mathbf{1} - \mathbf{a}_{\mathbf{v}_n})$$

which shifts l_n by a constant in each round. One can verify that $l_n(x) - l_n(x') = h_n(x) - h_n(x')$, for all x, x'. As the definition of regret is relative, we may work with h_n in online learning and use it to define the feedback.

The reason for using h_n instead of l_n is to make $\nabla_{\boldsymbol{\mu}} h_n((\mathbf{v}, \boldsymbol{\mu}))$ (and its unbiased approximation) a positive vector (because $\kappa \geq \|\mathbf{a}_{\mathbf{v}}\|_\infty$ for any $\mathbf{v} \in \mathcal{V}$), so that the regret bound can have a better dependency on the dimension for learning $\boldsymbol{\mu}$ that lives in the simplex $\mathcal{M}$. This is a common trick used in the online learning, e.g. in EXP3.

To run mirror descent, we set the first-order feedback g_n received by the learner as an unbiased estimate of $\nabla h_n(x_n)$. For round n, we construct g_n based on *two* calls of the generative model:

$$g_n = \begin{bmatrix} \mathbf{g}_{n,v} \\ \mathbf{g}_{n,\mu} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{p}}_n + \frac{1}{1-\gamma}(\gamma \tilde{\mathbf{P}}_n - \mathbf{E}_n)^\top \tilde{\boldsymbol{\mu}}_n \\ K(\kappa \hat{\mathbf{1}}_n - \hat{\mathbf{r}}_n - \frac{1}{1-\gamma}(\gamma \hat{\mathbf{P}}_n - \hat{\mathbf{E}}_n)\mathbf{v}_n) \end{bmatrix}$$

For $\mathbf{g}_{n,v}$, we sample $\mathbf{p}$, then sample $\boldsymbol{\mu}_n$ to get a state-action pair, and finally query the transition dynamics $\mathbf{P}$ at the state-action pair sampled from $\boldsymbol{\mu}_n$. ($\tilde{\mathbf{p}}_n$, $\tilde{\mathbf{P}}_n$, and $\tilde{\boldsymbol{\mu}}_n$ denote the single-sample empirical approximation of these probabilities.) For $\mathbf{g}_{n,\mu}$, we first sample *uniformly* a state-action pair (which explains the factor K), and then query the reward $\mathbf{r}$ and the transition dynamics $\mathbf{P}$. ($\hat{\mathbf{1}}_n$, $\hat{\mathbf{r}}_n$, $\hat{\mathbf{P}}_n$, and $\hat{\mathbf{E}}_n$ denote the single-sample empirical estimates.) To emphasize, we use˜and ̂to distinguish the empirical quantities obtained by these two independent queries. By construction, we have $\mathbf{g}_{n,\mu} \geq 0$. It is clear that this direction g_n is unbiased, i.e. $\mathbb{E}[g_n] = \nabla h_n(x_n)$. Moreover, it is extremely sparse and can be computed using $O(1)$ sample, computational, and memory complexities.

Let $y_N^* = (\mathbf{v}^{\hat{\pi}_N}, \boldsymbol{\mu}^*)$. We bound the regret by the following rearrangement.

$$
\begin{aligned}
&\mathrm{regret}_N(y_N^*) && (9.30)\\
&= \sum_{n=1}^N l_n(x_n) - \sum_{n=1}^N l_n(y_N^*)\\
&= \sum_{n=1}^N h_n(x_n) - \sum_{n=1}^N h_n(y_N^*)\\
&= \sum_{n=1}^N \nabla h_n(x_n)^\top (x_n - y_N^*)\\
&= \left(\sum_{n=1}^N (\nabla h_n(x_n) - g_n)^\top x_n\right) + \left(\sum_{n=1}^N g_n^\top (x_n - y_N^*)\right) + \left(\sum_{n=1}^N (g_n - \nabla h_n(x_n))^\top y_N^*\right)\\
&\le \left(\sum_{n=1}^N (\nabla h_n(x_n) - g_n)^\top x_n\right) + \left(\max_{x\in\mathcal{X}} \sum_{n=1}^N g_n^\top (x_n - x)\right) + \left(\sum_{n=1}^N (g_n - \nabla h_n(x_n))^\top y_N^*\right) && (9.31)
\end{aligned}
$$

We recognize the first term is a martingale $M_N = \sum_{n=1}^N (\nabla l_n(x_n) - g_n)^\top x_n$, because x_n does not depend on g_n. Therefore, we can appeal to standard martingale concentration property. For the second term, it can be upper bounded by standard regret analysis of mirror descent, by treating $g_n^\top x$ as the per-round loss. For the third term, because $y_N^* = (\mathbf{v}^{\hat{\pi}_N}, \boldsymbol{\mu}^*)$ depends on $\{g_n\}_{n=1}^N$, it is not a martingale. Nonetheless, we will be able to handle it through a union bound. Below we give details for bounding these three terms.

9.D.1 The First Term: Martingale Concentration

For the first term, $\sum_{n=1}^N (\nabla h_n(x_n) - g_n)^\top x_n$, we use a martingale concentration property. Specifically, we adopt a Bernstein-type inequality (McDiarmid, 1998, Theorem 3.15)

Lemma 9.D.1. (McDiarmid, 1998, Theorem 3.15) *Let $M_0, \ldots, M_N$ be a martingale and let $F_0 \subseteq F_1 \subseteq \cdots \subseteq F_n$ be the filtration such that $M_n = \mathbb{E}_{|F_n}[M_{n+1}]$. Suppose there are $b, \sigma < \infty$ such that for all n, given F_{n-1}, $M_n - M_{n-1} \le b$, and $\mathbb{V}_{|F_{n-1}}[M_n - M_{n-1}] \le \sigma^2$*

almost surely. Then for any $\epsilon \geq 0$*,*

$$P(M_N - M_0 \geq \epsilon) \leq \exp\left(\frac{-\epsilon^2}{2N\sigma^2(1+\frac{b\epsilon}{3N\sigma^2})}\right)$$

Lemma 9.D.1 implies, with probability at least $1-\delta$,

$$M_N - M_0 \leq \sqrt{2N\sigma^2(1+o(1))\log\left(\frac{1}{\delta}\right)}$$

where $o(1)$ means convergence to 0 as $N \to \infty$.

To apply Lemma 9.D.1, we need to provide bounds on the properties of the martingale difference

$$\begin{aligned} M_n - M_{n-1} &= (\nabla h_n(x_n) - g_n)^\top x_n \\ &= (\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}_n + (\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}_n \end{aligned}$$

For the first term $(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}_n$, we use the lemma below

Lemma 9.D.2. *Let* $\boldsymbol{\mu} \in \mathcal{M}$ *be arbitrary that is chosen independent of the randomness of* $\mathbf{g}_{n,\mu}$ *when* F_{n-1} *is given. Then it holds* $|(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}| \leq \frac{2(1+K)}{1-\gamma}$ *and* $\mathbb{V}_{|F_{n-1}}[(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}] \leq \frac{4K}{(1-\gamma)^2}$.

Proof. By triangular inequality,

$$|(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}| \leq |(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n})^\top \boldsymbol{\mu}| + |\mathbf{g}_{n,\mu}^\top \boldsymbol{\mu}|$$

For the deterministic part, using Lemma 9.C.3 and Hölder's inequality,

$$|(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n})^\top \boldsymbol{\mu}| \leq \kappa + \|\mathbf{a}_{\mathbf{v}_n}\|_\infty \|\boldsymbol{\mu}\|_1 \leq \frac{2}{1-\gamma}$$

For the stochastic part, let i_n be index of the sampled state-action pair and j_n be the index

of the transited state sampled at the pair given by i_n. With abuse of notation, we will use i_n to index both $\mathcal{S} \times \mathcal{A}$ and $\mathcal{S}$. With this notation, we may derive

$$\begin{aligned}
|\mathbf{g}_{n,\mu}^\top \boldsymbol{\mu}| &= |K\boldsymbol{\mu}^\top(\kappa\hat{\mathbf{1}}_n - \hat{\mathbf{r}}_n - \frac{1}{1-\gamma}(\gamma\hat{\mathbf{P}}_n - \hat{\mathbf{E}}_n)\mathbf{v}_n)| \\
&= K\mu_{i_n}|\kappa - r_{i_n} - \frac{\gamma v_{n,j_n} - v_{n,i_n}}{1-\gamma}| \\
&\leq \frac{2K\mu_{i_n}}{1-\gamma} \leq \frac{2K}{1-\gamma}
\end{aligned}$$

where we use the facts $r_{i_n}, v_{n,j_n}, v_{n,i_n} \in [0,1]$ and $\mu_{i_n} \leq 1$.

For $\mathbb{V}_{|F_{n-1}}[(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}_n]$, we can write it as

$$\begin{aligned}
\mathbb{V}_{|F_{n-1}}[(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}] &= \mathbb{V}_{|F_{n-1}}[\mathbf{g}_{n,\mu}^\top \boldsymbol{\mu}] \\
&\leq \mathbb{E}_{|F_{n-1}}[|\mathbf{g}_{n,\mu}^\top \boldsymbol{\mu}_n|^2] \\
&= \sum_{i_n} \frac{1}{K} \mathbb{E}_{j_n|i_n}\left[K^2\mu_{i_n}^2\left(\kappa - r_{i_n} - \frac{\gamma v_{n,j_n} - v_{n,i_n}}{1-\gamma}\right)^2\right] \\
&\leq \frac{4K}{(1-\gamma)^2}\sum_{i_n}\mu_{i_n}^2 \\
&\leq \frac{4K}{(1-\gamma)^2}\left(\sum_{i_n}\mu_{i_n}\right)^2 \leq \frac{4K}{(1-\gamma)^2}
\end{aligned}$$

where in the second inequality we use the fact that $|\kappa - r_{i_n} - \frac{\gamma v_{n,j_n} - v_{n,i_n}}{1-\gamma}| \leq \frac{2}{1-\gamma}$. ■

For the second term $(\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}_n$, we use this lemma.

Lemma 9.D.3. *Let $\mathbf{v} \in \mathcal{V}$ be arbitrary that is chosen independent of the randomness of $\mathbf{g}_{n,v}$ when F_{n-1} is given.. Then it holds $|(\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}| \leq \frac{4}{1-\gamma}$ and $\mathbb{V}_{|F_{n-1}}[(\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}] \leq \frac{4}{(1-\gamma)^2}$.*

Proof. We appeal to Lemma 9.C.3, which shows $\|\mathbf{b}_{\boldsymbol{\mu}_n}\|_1, \|\mathbf{g}_{n,v}\|_1 \leq \frac{2}{1-\gamma}$, and derive

$$|(\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}| \leq (\|\mathbf{b}_{\boldsymbol{\mu}_n}\|_1 + \|\mathbf{g}_{n,v}\|_1)\|\mathbf{v}\|_\infty \leq \frac{4}{1-\gamma}$$

Similarly, for the variance, we can write

$$\mathbb{V}_{|F_{n-1}}[(\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}] = \mathbb{V}_{|F_{n-1}}[\mathbf{g}_{n,v}^\top \mathbf{v}] \leq \mathbb{E}_{|F_{n-1}}[(\mathbf{g}_{n,v}^\top \mathbf{v})^2] \leq \frac{4}{(1-\gamma)^2} \qquad \blacksquare$$

Thus, with helps from the two lemmas above, we are able to show

$$M_n - M_{n-1} \leq |(\kappa \mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}_n| + |(\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}_n| \leq \frac{4 + 2(1+K)}{1-\gamma}$$

as well as (because $\mathbf{g}_{n,\mu}$ and $\mathbf{g}_{n,b}$ are computed using independent samples)

$$\begin{aligned}\mathbb{V}_{|F_{n-1}}[M_n - M_{n-1}] &= \mathbb{E}_{|F_{n-1}}[|(\kappa \mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu}_n|^2] + \mathbb{E}_{|F_{n-1}}[|(\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}_n|^2] \\ &\leq \frac{4(1+K)}{(1-\gamma)^2}\end{aligned}$$

Now, since $M_0 = 0$, by martingale concentration in Lemma 9.D.1, we have

$$P\left(\sum_{n=1}^{N} (\nabla h_n(x_n) - g_n)^\top x_n > \epsilon\right) \leq \exp\left(\frac{-\epsilon^2}{2N\sigma^2(1 + \frac{b\epsilon}{3N\sigma^2})}\right)$$

with $b = \frac{6+K}{1-\gamma}$ and $\sigma^2 = \frac{4(1+K)}{(1-\gamma)^2}$. This implies that, with probability at least $1-\delta$, it holds

$$\sum_{n=1}^{N} (\nabla h_n(x_n) - g_n)^\top x_n \leq \sqrt{N \frac{8(1+K)}{(1-\gamma)^2}(1+o(1)) \log\left(\frac{1}{\delta}\right)} = \tilde{O}\left(\frac{\sqrt{NK \log(\frac{1}{\delta})}}{1-\gamma}\right)$$

9.D.2 Static Regret of Mirror Descent

Next we move onto deriving the regret bound of mirror descent with respect to the online loss sequence:

$$\max_{x \in \mathcal{X}} \sum_{n=1}^{N} g_n^\top (x_n - x)$$

This part is quite standard; nonetheless, we provide complete derivations below.

We first recall a basic property of mirror descent

Lemma 9.D.4. *Let $\mathcal{X}$ be a convex set. Suppose R is 1-strongly convex with respect to some norm $\|\cdot\|$. Let g be an arbitrary vector and define, for $x \in \mathcal{X}$,*

$$y = \arg\min_{x' \in \mathcal{X}} \langle g, x' \rangle + B_R(x'||x)$$

Then for all $z \in \mathcal{X}$,

$$\langle g, y - z \rangle \leq B_R(z||x) - B_R(z||y) - B_R(y||x) \tag{9.32}$$

Proof. Recall the definition $B_R(x'||x) = R(x') - R(x) - \langle \nabla R(x), x' - x \rangle$. The optimality of the proximal map can be written as

$$\langle g + \nabla R(y) - \nabla R(x), y - z \rangle \leq 0, \qquad \forall z \in \mathcal{X}$$

By rearranging the terms, we can rewrite the above inequality in terms Bregman divergences as follows and derive the first inequality (10.37):

$$\begin{aligned}
\langle g, y - z \rangle &\leq \langle \nabla R(x) - \nabla R(y), y - z \rangle \\
&= B_R(z||x) - B_R(z||y) + \langle \nabla R(x) - \nabla R(y), y \rangle - \langle \nabla R(x), x \rangle \\
&\quad + \langle \nabla R(y), y \rangle + R(x) - R(y) \\
&= B_R(z||x) - B_R(z||y) + \langle \nabla R(x), y - x \rangle + R(x) - R(y) \\
&= B_R(z||x) - B_R(z||y) - B_R(y||x)
\end{aligned}$$

The second inequality is the consequence of (10.37). First, we rewrite (10.37) as

$$\langle g, x - z \rangle = B_R(z||x) - B_R(z||y) - B_R(y||x) + \langle g, x - y \rangle \qquad \blacksquare$$

Let $x' \in \mathcal{X}$ be arbitrary. Applying this lemma to the nth iteration of mirror descent in (10.7), we get

$$\langle g_n, x_{n+1} - x' \rangle \leq \frac{1}{\eta} \left(B_R(x'||x_n) - B_R(x'||x_{n+1}) - B_R(x_{n+1}||x_n) \right)$$

By a telescoping sum, we then have

$$\sum_{n=1}^{N} \langle g_n, x_n - x' \rangle \leq \frac{1}{\eta} B_R(x'||x_1) + \sum_{n=1}^{N} \langle g_n, x_{n+1} - x_n \rangle - \frac{1}{\eta} B_R(x_{n+1}||x_n)$$

We bound the right-hand side as follows. Recall, based on the geometry of $\mathcal{X} = \mathcal{V} \times \mathcal{M}$, we considered a natural Bregman divergence in the form

$$B_R(x'||x) = \frac{1}{2|\mathcal{S}|} \|\mathbf{v}' - \mathbf{v}\|_2^2 + KL(\boldsymbol{\mu}'||\boldsymbol{\mu})$$

Let $x_1 = (\mathbf{v}_1, \boldsymbol{\mu}_1)$ where $\boldsymbol{\mu}_1$ is uniform. By this choice, we have

$$\frac{1}{\eta} B_R(x'||x_1) \leq \frac{1}{\eta} \max_{x \in \mathcal{X}} B_R(x||x_1) \leq \frac{1}{\eta} \left(\frac{1}{2} + \log(K) \right)$$

For each item in the above sum, we decompose it as

$$\begin{aligned} \langle g_n, x_{n+1} - x_n \rangle - \frac{1}{\eta} B_R(x_{n+1}||x_n) = &\left(\mathbf{g}_{n,v}^\top (\mathbf{v}_{n+1} - \mathbf{v}_n) - \frac{1}{2\eta|\mathcal{S}|} \|\mathbf{v}_n - \mathbf{v}_{n+1}\|_2^2 \right) \\ &+ \left(\mathbf{g}_{n,\mu}^\top (\boldsymbol{\mu}_{n+1} - \boldsymbol{\mu}_n) + \frac{1}{\eta} KL(\boldsymbol{\mu}_{n+1}||\boldsymbol{\mu}_n) \right) \end{aligned}$$

and we upper bound them using the two lemmas below (recall $\mathbf{g}_{n,\mu} \geq 0$ due to the added $\kappa \mathbf{1}$ term).

Lemma 9.D.5. *For any vector x, y, g and scalar $\eta > 0$, it holds $\langle g, x - y \rangle + \frac{1}{2\eta} \|x - y\|_2^2 \leq \frac{\eta \|g\|_2^2}{2}$.*

Proof. By Cauchy-Swartz inequality, $\langle g, x - y \rangle + \frac{1}{2\eta} \|x - y\|_2^2 \leq \|g\|_2 \|x - y\|_2 + \frac{1}{2\eta} \|x -$

$y\|_2^2 \leq \frac{\eta\|g\|_2^2}{2}$ ■

Lemma 9.D.6. *Suppose $B_R(x||y) = KL(x||y)$ and x, y are probability distributions, and $g \geq 0$ elementwisely. For $\eta > 0$, $-\frac{1}{\eta}B_R(y||x) + \langle g, x - y\rangle \leq \frac{\eta}{2}\sum_i x_i(g_i)^2 = \frac{\eta}{2}\|g\|_x^2$.*

Proof. Let Δ denotes the unit simplex.

$$
\begin{aligned}
&- B_R(y||x) + \langle \eta g, x - y\rangle \\
&\leq \langle \eta g, x\rangle + \max_{y'\in\Delta} \langle -\eta g, y\rangle - B_R(y'||x) \\
&= \langle \eta g, x\rangle + \log\left(\sum_i x_i \exp(-\eta g_i)\right) && (\because \text{convex conjugate of KL divergence}) \\
&\leq \langle \eta g, x\rangle + \log\left(\sum_i x_i \left(1 - \eta g_i + \frac{1}{2}(\eta g_i)^2\right)\right) && (\because e^x \leq 1 + x + \frac{1}{2}x^2 \text{ for } x \leq 0) \\
&= \langle \eta g, x\rangle + \log\left(1 + \sum_i x_i \left(-\eta g_i + \frac{1}{2}(\eta g_i)^2\right)\right) \\
&\leq \langle \eta g, x\rangle + \sum_i x_i \left(-\eta g_i + \frac{1}{2}(\eta g_i)^2\right) && (\because \log(x) \leq x - 1) \\
&= \frac{1}{2}\sum_i x_i(\eta g_i)^2 = \frac{\eta^2}{2}\|g\|_x^2
\end{aligned}
$$

Finally, dividing both sides by η, we get the desired result. ■

Thus, we have the upper bound $\langle g_n, x_{n+1} - x_n\rangle - \frac{1}{\eta}B_R(x_{n+1}||x_n) = \frac{\eta|\mathcal{S}|\|\mathbf{g}_{n,v}\|_2^2}{2} + \frac{\eta\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2}{2}$. Together with the upper bound on $\frac{1}{\eta}B_R(x'||x_1)$, it implies that

$$
\begin{aligned}
\sum_{n=1}^N \langle g_n, x_n - x'\rangle &\leq \frac{1}{\eta}B_R(x'||x_1) + \sum_{n=1}^N \langle g_n, x_{n+1} - x_n\rangle - \frac{1}{\eta}B_R(x_{n+1}||x_n) \\
&\leq \frac{1}{\eta}\left(\frac{1}{2} + \log(K)\right) + \frac{\eta}{2}\sum_{n=1}^N |\mathcal{S}|\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2 \qquad (9.33)
\end{aligned}
$$

We can expect, with high probability, $\sum_{n=1}^N |\mathcal{S}|\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2$ concentrates toward

its expectation, i.e.

$$\sum_{n=1}^{N} |\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2 \leq \sum_{n=1}^{N} \mathbb{E}[|\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2] + o(N)$$

Below we quantify this relationship using martingale concentration. First we bound the expectation.

Lemma 9.D.7. $\mathbb{E}[\|\mathbf{g}_{n,v}\|_2^2] \leq \frac{4}{(1-\gamma)^2}$ *and* $\mathbb{E}[\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2] \leq \frac{4K}{(1-\gamma)^2}$.

Proof. For the first statement, using the fact that $\|\cdot\|_2 \leq \|\cdot\|_1$ and Lemma 9.C.3, we can write

$$\mathbb{E}[\|\mathbf{g}_{n,v}\|_2^2] \leq \mathbb{E}[\|\mathbf{g}_{n,v}\|_1^2] = \mathbb{E}[\|\tilde{\mathbf{p}}_n + \frac{1}{1-\gamma}(\gamma \tilde{\mathbf{P}}_n - \mathbf{E}_n)^\top \tilde{\boldsymbol{\mu}}_n\|_1^2] \leq \frac{4}{(1-\gamma)^2}$$

For the second statement, let i_n be index of the sampled state-action pair and j_n be the index of the transited state sampled at the pair given by i_n. With abuse of notation, we will use i_n to index both $\mathcal{S} \times \mathcal{A}$ and $\mathcal{S}$.

$$\begin{aligned}\mathbb{E}[\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2] &= \mathbb{E}\left[\sum_{i_n} \frac{1}{K} \mathbb{E}_{j_n|i_n}\left[K^2 \mu_{i_n}\left(\kappa - r_{i_n} - \frac{\gamma v_{n,j_n} - v_{n,i_n}}{1-\gamma}\right)^2\right]\right] \\ &\leq \frac{4K}{(1-\gamma)^2}\mathbb{E}\left[\sum_{i_n} \mu_{i_n}\right] \leq \frac{4K}{(1-\gamma)^2}\end{aligned}$$ ■

To bound the tail, we resort to the Höffding-Azuma inequality of martingale (McDiarmid, 1998, Theorem 3.14)

Lemma 9.D.8 (Azuma-Hoeffding)**.** *Let $M_0, \ldots, M_N$ be a martingale and let $F_0 \subseteq F_1 \subseteq \cdots \subseteq F_n$ be the filtration such that $M_n = \mathbb{E}_{|F_n}[M_{n+1}]$. Suppose there are $b < \infty$ such that for all n, given F_{n-1}, $|M_n - M_{n-1}| \leq b$. Then for any $\epsilon \geq 0$,*

$$P(M_N - M_0 \geq \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{Nb^2}\right)$$

To apply Lemma 9.D.8, we consider the martingale

$$M_N = \sum_{n=1}^{N} |\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2 - \left(\sum_{n=1}^{N} \mathbb{E}[|\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2] \right)$$

To bound the change of the size of martingale difference $|M_n - M_{n-1}|$, we again use similar steps in Lemma 9.D.7.

Lemma 9.D.9. $\|\mathbf{g}_{n,v}\|_2^2 \leq \frac{4}{(1-\gamma)^2}$ *and* $\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2 \leq \frac{4K^2}{(1-\gamma)^2}$.

Note $\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}}^2$ is K-factor larger than $\mathbb{E}[\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}}^2])$ and $K \geq 1$. Therefore, we have

$$|M_n - M_{n-1}| \leq |\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2 + |\mathcal{S}| \mathbb{E}[\|\mathbf{g}_{n,v}\|_2^2] + \mathbb{E}[\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2] \leq \frac{8(|\mathcal{S}| + K^2)}{(1-\gamma)^2}$$

Combining these results, we have, with probability as least $1 - \delta$,

$$\begin{aligned}
\sum_{n=1}^{N} |\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2 &\leq \sum_{n=1}^{N} \mathbb{E}[|\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2] + \frac{4\sqrt{2}(|\mathcal{S}| + K^2)}{(1-\gamma)^2} \sqrt{N \log\left(\frac{1}{\delta}\right)} \\
&\leq \frac{4(K + |\mathcal{S}|)}{(1-\gamma)^2} N + \frac{4\sqrt{2}(|\mathcal{S}| + K^2)}{(1-\gamma)^2} \sqrt{N \log\left(\frac{1}{\delta}\right)}
\end{aligned}$$

Now we suppose we set $\eta = \frac{1-\gamma}{\sqrt{KN}}$. From (9.40), we then have

$$\begin{aligned}
&\sum_{n=1}^{N} \langle g_n, x_n - x' \rangle \\
&\leq \frac{1}{\eta} \left(\frac{1}{2} + \log(K) \right) + \frac{\eta}{2} \sum_{n=1}^{N} |\mathcal{S}| \|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\mu}_n}^2 \\
&\leq \frac{\sqrt{KN}}{1-\gamma} \left(\frac{1}{2} + \log(K) \right) + \frac{1-\gamma}{\sqrt{KN}} \left(\frac{2(K + |\mathcal{S}|)}{(1-\gamma)^2} N + \frac{2\sqrt{2}(|\mathcal{S}| + K^2)}{(1-\gamma)^2} \sqrt{N \log\left(\frac{1}{\delta}\right)} \right) \\
&\leq \tilde{O}\left(\frac{\sqrt{KN}}{1-\gamma} + \frac{\sqrt{K^3 \log \frac{1}{\delta}}}{1-\gamma} \right)
\end{aligned}$$

9.D.3 Union Bound

Lastly, we provide an upper bound on the last component

$$\sum_{n=1}^{N}(g_n - \nabla h_n(x_n))^\top y_N^*$$

Because y_N^* depends on g_n, this term does not follow martingale concentration like the first component $\sum_{n=1}^{N}(\nabla h_n(x_n) - g_n)^\top x_n$ which we analyzed in Section 9.D.1 To resolve this issue, we utilize the concept of covering number and derive an union bound.

Recall for a compact set $\mathcal{Z}$ in a norm space, the covering number $\mathcal{N}(\mathcal{Z}, \epsilon)$ with $\epsilon > 0$ is the minimal number of ϵ-balls that covers $\mathcal{Z}$. That is, there is a set $\{z_i \in \mathcal{Z}\}_{i=1}^{\mathcal{N}(\mathcal{Z},\epsilon)}$ such that $\max_{z\in\mathcal{Z}} \min_{z'\in B(\mathcal{Z},\epsilon)} \|z - z'\| \le \epsilon$. Usually the covering number $\mathcal{N}(\mathcal{Z}, \epsilon)$ is polynomial in ϵ and perhaps exponential in the ambient dimension of $\mathcal{Z}$.

The idea of covering number can be used to provide an union bound of concentration over compact set, which we summarize as a lemma below.

Lemma 9.D.10. *Let f, g be two random L-Lipschitz functions. Suppose for some $a > 0$ and some fixed $z \in \mathcal{Z}$ selected independent of f, g, it holds*

$$P\left(|f(z) - f(z)| > \epsilon\right) \le \exp\left(-a\epsilon^2\right)$$

Then it holds that

$$P\left(\sup_{z\in\mathcal{Z}} |f(z) - g(z)| > \epsilon\right) \le \mathcal{N}\left(\mathcal{Z}, \frac{\epsilon}{4L}\right) \exp\left(\frac{-a\epsilon^2}{4}\right)$$

Proof. Let $\mathcal{C}$ denote a set of covers of size $\mathcal{N}(\mathcal{Z}, \frac{\epsilon}{4L})$ Then, for any $z \in \mathcal{Z}$ which could depend on f, g,

$$|f(z) - g(z)| \le \min_{z'\in\mathcal{C}} |f(z) - f(z')| + |f(z') - g(z')| + |g(z') - g(z)|$$

$$\leq \min_{z' \in \mathcal{C}} 2L\|z - z'\| + |f(z') - g(z')]|$$
$$\leq \frac{\epsilon}{2} + \max_{z' \in \mathcal{C}} |f(z') - g(z')|$$

Thus, $\sup_{z \in \mathcal{Z}} |f(z) - g(z)| > \epsilon \implies \max_{z' \in \mathcal{C}} |f(z') - g(z')| > \frac{\epsilon}{2}$. Therefore, we have the union bound.

$$P\left(\sup_{z \in \mathcal{Z}} |f(z) - \mathbb{E}[f(z)]| > \epsilon\right) \leq \mathcal{N}\left(\mathcal{Z}, \frac{\epsilon}{4L}\right) \exp\left(\frac{-a\epsilon^2}{4}\right)$$

■

We now use Lemma 9.D.10 to bound the component $\sum_{n=1}^{N} (g_n - \nabla h_n(x_n))^\top y_N^*$. We recall by definition, for $x = (\mathbf{v}, \boldsymbol{\mu})$,

$$(\nabla h_n(x_n) - g_n)^\top x = (\kappa \mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\mu} + (\mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \mathbf{v}$$

Because $y_N^* = (\mathbf{v}^{\hat{\pi}_N}, \boldsymbol{\mu}^*)$, we can write the sum of interest as

$$\sum_{n=1}^{N} (g_n - \nabla h_n(x_n))^\top y_N^* = \sum_{n=1}^{N} (\mathbf{g}_{n,\mu} - \kappa \mathbf{1} + \mathbf{a}_{\mathbf{v}_n})^\top \boldsymbol{\mu}^* + \sum_{n=1}^{N} (\mathbf{g}_{n,v} - \mathbf{b}_{\boldsymbol{\mu}_n})^\top \mathbf{v}^{\hat{\pi}_N}$$

For the first term, because $\boldsymbol{\mu}^*$ is set beforehand by the MDP definition and does not depend on the randomness during learning, it is a martingale and we can apply the steps in Section 9.D.1 to show,

$$\sum_{n=1}^{N} (\mathbf{g}_{n,\mu} - \kappa \mathbf{1} + \mathbf{a}_{\mathbf{v}_n})^\top \boldsymbol{\mu}^* = \tilde{O}\left(\frac{\sqrt{NK \log(\frac{1}{\delta})}}{1 - \gamma}\right)$$

For the second term, because $\mathbf{v}^{\hat{\pi}_N}$ depends on the randomness in the learning process, we need to use an union bound. Following the steps in Section 9.D.1, we see that for some

fixed $\mathbf{v} \in \mathcal{V}$, it holds

$$\sum_{n=1}^{N} (\mathbf{g}_{n,v} - \mathbf{b}_{\boldsymbol{\mu}_n})^\top \mathbf{v} = \tilde{O}\left(\frac{\sqrt{N \log(\frac{1}{\delta})}}{1-\gamma}\right)$$

(Note it does not have the $\sqrt{K}$ factor because of Lemma 9.D.3). To apply Lemma 9.D.10, we need to know the order of covering number of $\mathcal{V}$. Since $\mathcal{V}$ is an $|\mathcal{S}|$-dimensional unit cube in the positive orthant, it is straightforward to show (by simply discretizing evenly in each dimension) $\mathcal{N}(\mathcal{V}, \epsilon) \leq (1/\epsilon)^{|\mathcal{S}|}$. This would imply that

$$\sup_{v \in \mathcal{V}} \sum_{n=1}^{N} (\mathbf{g}_{n,v} - \mathbf{b}_{\boldsymbol{\mu}_n})^\top \mathbf{v} = \epsilon = \tilde{O}\left(\frac{\sqrt{N \log(\frac{\mathcal{N}(\mathcal{V},\epsilon)}{\delta})}}{1-\gamma}\right) = \tilde{O}\left(\frac{\sqrt{N|\mathcal{S}| \log(\frac{1}{\delta})}}{1-\gamma}\right)$$

Combining these two steps, we have shown overall, with probability at least $1-\delta$,

$$\sum_{n=1}^{N} (g_n - \nabla h_n(x_n))^\top y_N^* = \tilde{O}\left(\frac{\sqrt{NK \log(\frac{1}{\delta})}}{1-\gamma}\right).$$

9.D.4 Summary

In the previous subsections, we have provided high probability upper bounds for each term in the decomposition

$$\text{regret}_N(y_N^*) \leq \left(\sum_{n=1}^{N} (\nabla h_n(x_n) - g_n)^\top x_n\right) + \left(\max_{x \in \mathcal{X}} \sum_{n=1}^{N} g_n^\top (x_n - x)\right) + \left(\sum_{n=1}^{N} (g_n - \nabla h_n(x_n))^\top y_N^*\right)$$

implying with probability at least $1-\delta$,

$$\text{regret}_N(y_N^*) \leq \tilde{O}\left(\frac{\sqrt{NK \log(\frac{1}{\delta})}}{1-\gamma}\right) + \tilde{O}\left(\frac{\sqrt{KN}}{1-\gamma} + \frac{\sqrt{K^3 \log \frac{1}{\delta}}}{1-\gamma}\right) = \tilde{O}\left(\frac{\sqrt{N|\mathcal{S}||\mathcal{A}| \log(\frac{1}{\delta})}}{1-\gamma}\right)$$

By Theorem 9.4.1, this would imply with probability at least $1-\delta$,

$$V^{\hat{\pi}_N}(p) \geq V^*(p) - \frac{\text{regret}_N(y_N^*)}{N} \geq V^*(p) - \tilde{O}\left(\frac{\sqrt{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}}{(1-\gamma)\sqrt{N}}\right)$$

In other words, the sample complexity of mirror descent to obtain an ϵ approximately optimal policy (i.e. $V^*(p) - V^{\hat{\pi}_N}(p) \leq \epsilon$) is at most $\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|\log(\frac{1}{\delta})}{(1-\gamma)^2\epsilon^2}\right)$.

9.E Sample Complexity of Mirror Descent with Basis Functions

Here we provide further discussions on the sample complexity of running Algorithm 3 with linearly parameterized function approximators and the proof of Theorem 9.5.2.

Theorem 9.5.2. *Under a proper choice of Θ and B_R, with probability $1-\delta$, Algorithm 3 learns an $(\epsilon + \epsilon_{\Theta,N})$-optimal policy with $\tilde{O}\left(\frac{dim(\Theta)\log(\frac{1}{\delta})}{(1-\gamma)^2\epsilon^2}\right)$ samples.*

9.E.1 Setup

We suppose that the decision variable is parameterized in the form $x_\theta = (\boldsymbol{\Phi}\boldsymbol{\theta}_v, \boldsymbol{\Psi}\boldsymbol{\theta}_\mu)$, where $\boldsymbol{\Phi}, \boldsymbol{\Psi}$ are nonlinear basis functions and $(\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu) \in \Theta$ are the parameters to learn. For modeling the value function, we suppose each column in $\boldsymbol{\Phi}$ is a vector (i.e. function) such that its $\|\cdot\|_\infty$ is less than one. For modeling the state-action distribution, we suppose each column in $\boldsymbol{\Psi}$ is a valid state-action distribution from which we can draw samples. In other words, every column of $\boldsymbol{\Phi}$ belongs to $\mathcal{V}$, and every column of $\boldsymbol{\Psi}$ belongs to $\mathcal{M}$.

Considering the geometry of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, we consider a compact and convex parameter set

$$\Theta = \{\theta = (\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu) : \|\boldsymbol{\theta}_v\|_2 \leq \frac{C_v}{\sqrt{dim(\boldsymbol{\theta}_v)}}, \boldsymbol{\theta}_\mu \geq 0, \|\boldsymbol{\theta}_\mu\|_1 = 1\}$$

where $C_v < \infty$. The constant C_v acts as a regularization in learning: if it is too small, the bias (captured as $\epsilon_{\Theta,N}$ in Corollary 9.4.1 restated below) becomes larger; if it is too large,

the learning becomes slower.

This choice of Θ makes sure, for $\theta = (\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu) \in \Theta$, $\boldsymbol{\Psi}\boldsymbol{\theta}_\mu \in \mathcal{M}$ and $\|\boldsymbol{\Phi}\boldsymbol{\theta}_v\|_\infty \leq \|\boldsymbol{\theta}_v\|_1 \leq C_v$. Therefore, by the above construction, we can verify that the requirement in Corollary 9.4.1 is satisfied, i.e. for $\theta = (\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu) \in \Theta$, we have $(\boldsymbol{\Phi}\boldsymbol{\theta}_v, \boldsymbol{\Psi}\boldsymbol{\theta}_\mu) \in \mathcal{X}_\Theta$.

Corollary 9.4.1. *Let $X_N = \{x_n \in \mathcal{X}_\theta\}_{n=1}^N$ be any sequence. Let $\hat{\pi}_N$ be the policy given either by the average or the best decision in X_N. It holds that*

$$V^{\hat{\pi}_N}(p) \geq V^*(p) - \frac{\mathrm{regret}_N(\Theta)}{N} - \epsilon_{\Theta,N}$$

where $\epsilon_{\Theta,N} = \min_{x_\theta \in \mathcal{X}_\theta} r_{ep}(\hat{x}_N; y_N^*) - r_{ep}(\hat{x}_N; x_\theta)$ measures the expressiveness of X_θ, and $\mathrm{regret}_N(\Theta) \coloneqq \sum_{n=1}^N l_n(x_n) - \min_{x \in \mathcal{X}_\Theta} \sum_{n=1}^N l_n(x)$.

9.E.2 Online Loss and Sampled Gradient

Let $\theta = (\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu) \in \Theta$. In view of the parameterization above, we can identify the online loss in (9.23) in the parameter space as

$$h_n(\theta) \coloneqq \mathbf{b}_{\boldsymbol{\mu}_n}^\top \boldsymbol{\Phi}\boldsymbol{\theta}_v + \boldsymbol{\theta}_\mu^\top \boldsymbol{\Psi}^\top (\tfrac{1}{1-\gamma}\mathbf{1} - \mathbf{a}_{\mathbf{v}_n}) \tag{9.34}$$

where we have the natural identification $x_n = (\mathbf{v}_n, \boldsymbol{\mu}_n) = (\boldsymbol{\Phi}\boldsymbol{\theta}_{v,n}, \boldsymbol{\Psi}\boldsymbol{\theta}_{\mu,n})$ and $\theta_n = (\boldsymbol{\theta}_{v,n}, \boldsymbol{\theta}_{\mu,n}) \in \Theta$ are the decision made by the online learner in the nth round. For writing convenience, we will continue to overload h_n as a function of parameter θ in the following analyses.

Mirror descent requires gradient estimates of $\nabla h_n(\theta_n)$. Here we construct an unbiased stochastic estimate of $\nabla h_n(\theta_n)$ as

$$g_n = \begin{bmatrix} \mathbf{g}_{n,v} \\ \mathbf{g}_{n,\mu} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}^\top (\tilde{\mathbf{p}}_n + \frac{1}{1-\gamma}(\gamma \tilde{\mathbf{P}}_n - \mathbf{E}_n)^\top \tilde{\boldsymbol{\mu}}_n) \\ dim(\boldsymbol{\theta}_\mu) \hat{\boldsymbol{\Psi}}_n^\top (\frac{1}{1-\gamma}\hat{\mathbf{1}}_n - \hat{\mathbf{r}}_n - \frac{1}{1-\gamma}(\gamma \hat{\mathbf{P}}_n - \hat{\mathbf{E}}_n)\mathbf{v}_n) \end{bmatrix} \tag{9.35}$$

using two calls of the generative model (again we overload the symbol g_n for the analyses in this section):

- The upper part $\mathbf{g}_{n,v}$ is constructed similarly as before in (9.24): First we sample the initial state from the initial distribution, the state-action pair using the learned state-action distribution, and then then transited state at the sampled state-action pair. Then we evaluate $\mathbf{\Phi}$'s values at those samples to construct $\mathbf{g}_{n,v}$. Thus, $\mathbf{g}_{n,v}$ would generally be a dense vector of size $dim(\boldsymbol{\theta}_v)$ (unless the columns of $\mathbf{\Phi}$ are sparse to begin with).

- The lower part $\mathbf{g}_{n,\mu}$ is constructed slightly differently. Recall for the tabular version in (9.24), we uniformly sample over the state and action spaces. Here instead we first sample uniformly a column (i.e. a basis function) in $\mathbf{\Psi}$ and then sample a state-action pair according to the sampled column, which is a distribution by our choice. Therefore, the multiplier due to uniform sampling in the second row of (9.35) is now $dim(\boldsymbol{\theta}_\mu)$ rather than $|\mathcal{S}||\mathcal{A}|$ in (9.24). The matrix $\hat{\mathbf{\Psi}}_n$ is a extremely sparse, where only the single sampled entry (the column and the state-action pair) is one and the others are zero. In fact, one can think of the tabular version is simply using basis functions $\mathbf{\Psi} = \mathbf{I}$, i.e. each column is a delta distribution, and the expression in (9.35) matches the one in (9.24) under this identification.

It is straightforward to verify that $\mathbb{E}[g_n] = \nabla h_n(\theta_n)$ for g_n in (9.35).

9.E.3 Proof of Theorem 9.5.2

We follow the same steps of the analysis of the tabular version. We will highlight the differences/improvement due to using function approximations.

First, we use Corollary 9.4.1 in place of Theorem 9.4.1. To properly consider the randomness, we revisit its derivation to slightly tighten the statement Corollary 9.4.1, which

was simplified for the sake of cleaner exposition. Define

$$y^*_{N,\theta} = (\mathbf{v}^*_{N,\theta}, \boldsymbol{\mu}^*_\theta) \coloneqq \arg\max_{x_\theta \in \mathcal{X}_\theta} r_{ep}(\hat{x}_N; x_\theta)$$

For writing convenience, let us also denote $\theta^*_N = (\boldsymbol{\theta}^*_{v,N}, \boldsymbol{\theta}^*_\mu) \in \Theta$ as the corresponding parameter of $y^*_{N,\theta}$. We remark that $\boldsymbol{\mu}^*_\theta$ (i.e.$\boldsymbol{\theta}^*_\mu$), which tries to approximate $\boldsymbol{\mu}^*$, is fixed before the learning process, whereas $\mathbf{v}^*_{N,\theta}$ (i.e. $\boldsymbol{\theta}^*_{v,N}$) could depend on the stochasticity in the learning. Using this new notation and the steps in the proof of Corollary 9.4.1, we can write

$$\begin{aligned} V^*(p) - V^{\hat{\pi}_N}(p) &= r_{ep}(\hat{x}_N; y^*_N) \\ &= \epsilon_{\Theta,N} + r_{ep}(\hat{x}_N; y^*_{N,\theta}) \leq \epsilon_{\Theta,N} + \frac{\text{regret}_N(y^*_{N,\theta})}{N} \end{aligned}$$

where the first equality is Proposition 9.4.1, the last inequality follows the proof of Theorem 9.4.1, and we recall the definition $\epsilon_{\Theta,N} = r_{ep}(\hat{x}_N; y^*_N) - r_{ep}(\hat{x}_N; y^*_{N,\theta})$.

The rest of the proof is very similar to that of Theorem 9.4.1, because linear parameterization does not change the convexity of the loss sequence. Let $y^*_N = (\mathbf{v}^{\hat{\pi}_N}, \boldsymbol{\mu}^*)$. We bound the regret by the following rearrangement.

$$\begin{aligned} &\text{regret}_N(y^*_{N,\theta}) \qquad (9.36) \\ &= \sum_{n=1}^N l_n(x_n) - \sum_{n=1}^N l_n(y^*_{N,\theta}) \\ &= \sum_{n=1}^N h_n(\theta_n) - \sum_{n=1}^N h_n(\theta^*_N) \\ &= \sum_{n=1}^N \nabla h_n(\theta_n)^\top (\theta_n - \theta^*_N) \\ &= \left(\sum_{n=1}^N (\nabla h_n(\theta_n) - g_n)^\top \theta_n\right) + \left(\sum_{n=1}^N g_n^\top (\theta_n - \theta^*_N)\right) + \left(\sum_{n=1}^N (g_n - \nabla h_n(\theta_n))^\top \theta^*_N\right) \end{aligned}$$

$$\leq \left(\sum_{n=1}^{N}(\nabla h_n(\theta_n) - g_n)^\top \theta_n\right) + \left(\max_{\theta\in\Theta}\sum_{n=1}^{N} g_n^\top(\theta_n - \theta)\right) + \left(\sum_{n=1}^{N}(g_n - \nabla h_n(\theta_n))^\top \theta_N^*\right) \tag{9.37}$$

where the second equality is due to the identifcation in (9.34).

We will solve this online learning problem with mirror descent

$$\theta_{n+1} = \arg\min_{\theta\in\Theta} \langle g_n, \theta\rangle + \frac{1}{\eta} B_R(\theta||\theta_n) \tag{9.38}$$

with step size $\eta > 0$ and a Bregman divergence that is a straightforward extension of (9.22)

$$B_R(\theta'||\theta) = \tfrac{1}{2}\tfrac{dim(\theta_v)}{C_v^2}\|\boldsymbol{\theta}'_v - \boldsymbol{\theta}_v\|_2^2 + KL(\boldsymbol{\theta}'_\mu||\boldsymbol{\theta}_\mu) \tag{9.39}$$

where the constant $\frac{dim(\theta_v)}{C_v^2}$ is chosen to make the size of Bregman divergence dimension-free (at least up to $\log$ factors). Below we analyze the size of the three terms in (9.36) like what we did for Theorem 9.5.1.

9.E.4 The First Term: Martingale Concentration

The first term is a martingale. We will use this part to highlight the different properties due to using basis functions. The proof follows the steps in Section 9.D.1, but now the martingale difference of interest is instead

$$\begin{aligned} M_n - M_{n-1} &= (\nabla h_n(\theta_n) - g_n)^\top \theta_n \\ &= (\boldsymbol{\Psi}^\top(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n}) - \mathbf{g}_{n,\mu})^\top \boldsymbol{\theta}_{\mu,n} + (\boldsymbol{\Phi}^\top \mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \boldsymbol{\theta}_{v,n} \end{aligned}$$

They now have nicer properties due to the way $\mathbf{g}_{n,\mu}$ is sampled.

For the first term $(\boldsymbol{\Psi}^\top(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n}) - \mathbf{g}_{n,\mu})^\top \boldsymbol{\theta}_{\mu,n}$, we use the lemma below, where we recall the filtration F_n is naturally defined as $\{\theta_1, \ldots, \theta_n\}$.

Lemma 9.E.1. *Let $\theta = (\boldsymbol{\theta}_v, \boldsymbol{\theta}_\mu) \in \Theta$ be arbitrary that is chosen independent of the randomness of $\mathbf{g}_{n,\mu}$ when F_{n-1} is given. Then it holds $|(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\theta}| \leq \frac{2(1+dim(\boldsymbol{\theta}_{\boldsymbol{\mu}}))}{1-\gamma}$ and $\mathbb{V}_{|F_{n-1}}[(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\theta}_n] \leq \frac{4dim(\boldsymbol{\theta}_{\boldsymbol{\mu}})}{(1-\gamma)^2}$.*

Proof. By triangular inequality,

$$|(\boldsymbol{\Psi}^\top(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n}) - \mathbf{g}_{n,\mu})^\top \boldsymbol{\theta}_\mu| \leq |(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n})^\top \boldsymbol{\Psi}\boldsymbol{\theta}_\mu| + |\mathbf{g}_{n,\mu}^\top \boldsymbol{\theta}_\mu|$$

For the deterministic part, using Lemma 9.C.3 and Hölder's inequality,

$$|(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n})^\top \boldsymbol{\Psi}\boldsymbol{\theta}_\mu| \leq \kappa + \|\mathbf{a}_{\mathbf{v}_n}\|_\infty \|\boldsymbol{\Psi}\boldsymbol{\theta}_\mu\|_1 \leq \frac{2}{1-\gamma}$$

For the stochastic part, let k_n denote the sampled column index, i_n be index of the sampled state-action pair using the column of k_n, and j_n be the index of the transited state sampled at the pair given by i_n. With abuse of notation, we will use i_n to index both $\mathcal{S} \times \mathcal{A}$ and $\mathcal{S}$. Let $\boldsymbol{\mu} = \boldsymbol{\Psi}\boldsymbol{\theta}_\mu$. With this notation, we may derive

$$\begin{aligned}
|\mathbf{g}_{n,\mu}^\top \boldsymbol{\theta}_\mu| &= |dim(\boldsymbol{\theta}_\mu)\boldsymbol{\theta}_{\boldsymbol{\mu}}^\top \hat{\boldsymbol{\Psi}}_n^\top (\kappa\hat{\mathbf{1}}_n - \hat{\mathbf{r}}_n - \frac{1}{1-\gamma}(\gamma\hat{\mathbf{P}}_n - \hat{\mathbf{E}}_n)\mathbf{v}_n)| \\
&= dim(\boldsymbol{\theta}_\mu)\theta_{\mu,k_n}|\kappa - r_{i_n} - \frac{\gamma v_{n,j_n} - v_{n,i_n}}{1-\gamma}| \\
&\leq \frac{2dim(\boldsymbol{\theta}_\mu)\theta_{\mu,k_n}}{1-\gamma} \leq \frac{2dim(\boldsymbol{\theta}_\mu)}{1-\gamma}
\end{aligned}$$

where we use the facts $r_{i_n}, v_{n,j_n}, v_{n,i_n} \in [0,1]$ and $\theta_{\mu,k_n} \leq 1$.

Let $\boldsymbol{\psi}_\mu^{(k)}$ denote the kth column of $\boldsymbol{\Psi}$. For $\mathbb{V}_{|F_{n-1}}[(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n} - \mathbf{g}_{n,\mu})^\top \boldsymbol{\theta}_n]$, we can write it as

$$\begin{aligned}
&\mathbb{V}_{|F_{n-1}}[(\boldsymbol{\Psi}^\top(\kappa\mathbf{1} - \mathbf{a}_{\mathbf{v}_n}) - \mathbf{g}_{n,\mu})^\top \boldsymbol{\theta}_\mu] \\
&= \mathbb{V}_{|F_{n-1}}[\mathbf{g}_{n,\mu}^\top \boldsymbol{\theta}_n] \\
&\leq \mathbb{E}_{|F_{n-1}}[|\mathbf{g}_{n,\mu}^\top \boldsymbol{\theta}_n|^2]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{k_n} \frac{1}{dim(\boldsymbol{\theta}_\mu)} \sum_{i_n} \psi_{\mu,i_n}^{(k_n)} \mathbb{E}_{j_n|i_n} \left[dim(\boldsymbol{\theta}_\mu)^2 \theta_{\mu,k_n}^2 \left(\kappa - r_{i_n} - \frac{\gamma v_{n,j_n} - v_{n,i_n}}{1-\gamma} \right)^2 \right] \\
&\leq \frac{4dim(\boldsymbol{\theta}_\mu)}{(1-\gamma)^2} \sum_{k_n} \theta_{\mu,k_n}^2 \sum_{i_n} \psi_{\mu,i_n}^{(k_n)} \\
&\leq \frac{4dim(\boldsymbol{\theta}_\mu)}{(1-\gamma)^2} \left(\sum_{k_n} \theta_{\mu,k_n} \right)^2 \leq \frac{4dim(\boldsymbol{\theta}_\mu)}{(1-\gamma)^2}
\end{aligned}
$$

where in the second inequality we use the fact that $|\kappa - r_{i_n} - \frac{\gamma v_{n,j_n} - v_{n,i_n}}{1-\gamma}| \leq \frac{2}{1-\gamma}$. ■

For the second term $(\boldsymbol{\Phi}^\top \mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \boldsymbol{\theta}_{v,n}$, we use this lemma.

Lemma 9.E.2. *Let $\boldsymbol{\theta} \in \mathcal{V}$ be arbitrary that is chosen independent of the randomness of $\mathbf{g}_{n,v}$ when F_{n-1} is given. Then it holds $|(\boldsymbol{\Phi}^\top \mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \boldsymbol{\theta}| \leq \frac{4C_v}{1-\gamma}$ and $\mathbb{V}_{|F_{n-1}}[(\boldsymbol{\Phi}^\top \mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \boldsymbol{\theta}] \leq \frac{4C_v^2}{(1-\gamma)^2}$.*

Proof. We appeal to Lemma 9.C.3, which shows $\|\mathbf{b}_{\boldsymbol{\mu}_n}\|_1 \leq \frac{2}{1-\gamma}$ and

$$
\|\tilde{\mathbf{p}}_n + \frac{1}{1-\gamma}(\gamma \tilde{\mathbf{P}}_n - \mathbf{E}_n)^\top \tilde{\boldsymbol{\mu}}_n\|_1 \leq \frac{2}{1-\gamma}
$$

Therefore, overall we can derive

$$
|(\boldsymbol{\Phi}^\top \mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \boldsymbol{\theta}| \leq \left(\|\mathbf{b}_{\boldsymbol{\mu}_n}\|_1 \ + \|\tilde{\mathbf{p}}_n + \frac{1}{1-\gamma}(\gamma \tilde{\mathbf{P}}_n - \mathbf{E}_n)^\top \tilde{\boldsymbol{\mu}}_n\|_1 \right) \|\boldsymbol{\Phi} \boldsymbol{\theta}_v\|_\infty \leq \frac{4C_v}{1-\gamma}
$$

where we use again each column in $\boldsymbol{\Phi}$ has $\|\cdot\|_\infty$ less than one, and $\|\cdot\|_\infty \leq \|\cdot\|_2$. Similarly, for the variance, we can write

$$
\mathbb{V}_{|F_{n-1}}[(\boldsymbol{\Phi}^\top \mathbf{b}_{\boldsymbol{\mu}_n} - \mathbf{g}_{n,v})^\top \boldsymbol{\theta}] = \mathbb{V}_{|F_{n-1}}[\mathbf{g}_{n,v}^\top \boldsymbol{\theta}] \leq \mathbb{E}_{|F_{n-1}}[(\mathbf{g}_{n,v}^\top \boldsymbol{\theta})^2] \leq \frac{4C_v^2}{(1-\gamma)^2}
$$

■

From the above two lemmas, we see the main difference from the what we had in Section 9.D.1 for the tabular case is that, the martingale difference now scales in $O\left(\frac{C_v + dim(\boldsymbol{\theta}_{\boldsymbol{\mu}})}{1-\gamma}\right)$ instead of $O\left(\frac{|\mathcal{S}||\mathcal{A}|}{1-\gamma}\right)$, and its variance scales in $O\left(\frac{C_v^2 + dim(\boldsymbol{\theta}_{\boldsymbol{\mu}})}{(1-\gamma)^2}\right)$ instead of $O\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^2}\right)$. We note the constant C_v is universal, independent of the problem size.

Following the similar steps in Section 9.D.1, these new results imply that

$$P\left(\sum_{n=1}^{N}(\nabla h_n(\theta_n)-g_n)^\top\theta_n>\epsilon\right)\leq\exp\left(\frac{-\epsilon^2}{2N\sigma^2(1+\frac{b\epsilon}{3N\sigma^2})}\right)$$

with $b=O\left(\frac{C_v+dim(\boldsymbol{\theta_\mu})}{1-\gamma}\right)$ and $O\left(\frac{C_v^2+dim(\boldsymbol{\theta_\mu})}{(1-\gamma)^2}\right)$. This implies that, with probability at least $1-\delta$, it hold

$$\sum_{n=1}^{N}(\nabla h_n(\theta_n)-g_n)^\top\theta_n=\tilde{O}\left(\frac{\sqrt{N(C_v^2+dim(\boldsymbol{\theta_\mu}))\log(\frac{1}{\delta})}}{1-\gamma}\right)$$

9.E.5 Static Regret of Mirror Descent

Again the steps here are very similar to those in Section 9.D.2. We concern bounding the static regret.

$$\max_{\theta\in\Theta}\sum_{n=1}^{N}g_n^\top(\theta_n-\theta)$$

From Section 9.D.2, we recall this can be achieved by the mirror descent's optimality condition. The below inequality is true, for any $\theta'\in\Theta$:

$$\sum_{n=1}^{N}\langle g_n,\theta_n-\theta'\rangle\leq\frac{1}{\eta}B_R(\theta'||\theta_1)+\sum_{n=1}^{N}\langle g_n,\theta_{n+1}-\theta_n\rangle-\frac{1}{\eta}B_R(\theta_{n+1}||\theta_n)$$

Based on our choice of Bregman divergence given in (9.39), i.e.

$$B_R(\theta'||\theta)=\tfrac{1}{2}\tfrac{dim(\theta_v)}{C_v^2}\|\boldsymbol{\theta}'_v-\boldsymbol{\theta}_v\|_2^2+KL(\boldsymbol{\theta}'_\mu||\boldsymbol{\theta}_\mu),\tag{9.39}$$

we have $\frac{1}{\eta}B_R(\theta'||\theta_1)\leq\frac{\tilde{O}(1)}{\eta}$. For each $\langle g_n,\theta_{n+1}-\theta_n\rangle-\frac{1}{\eta}B_R(\theta_{n+1}||\theta_n)$, we will use again the two basic lemmas we proved in Section 9.D.2.

Lemma 9.D.5. *For any vector x,y,g and scalar $\eta>0$, it holds $\langle g,x-y\rangle+\frac{1}{2\eta}\|x-y\|_2^2\leq$*

$\frac{\eta\|g\|_2^2}{2}$.

Lemma 9.D.6. *Suppose $B_R(x||y) = KL(x||y)$ and x, y are probability distributions, and $g \geq 0$ elementwisely. For $\eta > 0$, $-\frac{1}{\eta}B_R(y||x) + \langle g, x - y\rangle \leq \frac{\eta}{2}\sum_i x_i(g_i)^2 = \frac{\eta}{2}\|g\|_x^2$.*

Thus, we have the upper bound

$$\langle g_n, \theta_{n+1} - \theta_n\rangle - \frac{1}{\eta}B_R(\theta_{n+1}||\theta_n) = \frac{C_v^2}{dim(\theta_v)}\frac{\eta\|\mathbf{g}_{n,v}\|_2^2}{2} + \frac{\eta\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2}{2}$$

Together with the upper bound on $\frac{1}{\eta}B_R(x'||x_1)$, it implies that

$$\begin{aligned}\sum_{n=1}^N \langle g_n, x_n - x'\rangle &\leq \frac{1}{\eta}B_R(x'||x_1) + \sum_{n=1}^N \langle g_n, x_{n+1} - x_n\rangle - \frac{1}{\eta}B_R(x_{n+1}||x_n)\\ &\leq \frac{\tilde{O}(1)}{\eta} + \frac{\eta}{2}\sum_{n=1}^N \frac{C_v^2}{dim(\theta_v)}\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2 \qquad (9.40)\end{aligned}$$

We can expect, with high probability, $\sum_{n=1}^N \frac{C_v^2}{dim(\theta_v)}\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2$ concentrates toward its expectation, i.e.

$$\sum_{n=1}^N \frac{C_v^2}{dim(\theta_v)}\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2 \leq \sum_{n=1}^N \mathbb{E}\left[\frac{C_v^2}{dim(\theta_v)}\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2\right] + o(N)$$

To bound the right-hand side, we will use the upper bounds below, which largely follow the proof of Lemma 9.E.1 and Lemma 9.E.2.

Lemma 9.E.3. *$\mathbb{E}[\|\mathbf{g}_{n,v}\|_2^2] \leq \frac{4dim(\boldsymbol{\theta}_v)}{(1-\gamma)^2}$ and $\mathbb{E}[\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2] \leq \frac{4dim(\boldsymbol{\theta}_\mu)}{(1-\gamma)^2}$.*

Lemma 9.E.4. *$\|\mathbf{g}_{n,v}\|_2^2 \leq \frac{4dim(\boldsymbol{\theta}_v)}{(1-\gamma)^2}$ and $\|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2 \leq \frac{4dim(\boldsymbol{\theta}_\mu)^2}{(1-\gamma)^2}$.*

By Azuma-Hoeffding's inequality in Lemma 9.D.8,

$$\begin{aligned}&\sum_{n=1}^N \frac{C_v^2}{dim(\theta_v)}\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\boldsymbol{\mu},n}}^2\\ &\leq \sum_{n=1}^N \mathbb{E}\left[\frac{C_v^2}{dim(\theta_v)}\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\boldsymbol{\mu},n}}^2\right] + O\left(\frac{C_v^2 + dim(\boldsymbol{\theta}_\mu)^2}{(1-\gamma)^2}\sqrt{N\log\left(\frac{1}{\delta}\right)}\right)\end{aligned}$$

$$\leq O\left(\frac{C_v^2 + dim(\boldsymbol{\theta}_\mu)}{(1-\gamma)^2}N\right) + O\left(\frac{C_v^2 + dim(\boldsymbol{\theta}_\mu)^2}{(1-\gamma)^2}\sqrt{N\log\left(\frac{1}{\delta}\right)}\right)$$

Now we suppose we set $\eta = O\left(\frac{1-\gamma}{\sqrt{N(C_v^2+dim(\boldsymbol{\theta}_\mu))}}\right)$. We have

$$\sum_{n=1}^{N}\langle g_n, \theta_n - \theta'\rangle \leq \frac{\tilde{O}(1)}{\eta} + \frac{\eta}{2}\sum_{n=1}^{N}\frac{C_v^2}{dim(\theta_v)}\|\mathbf{g}_{n,v}\|_2^2 + \|\mathbf{g}_{n,\mu}\|_{\boldsymbol{\theta}_{\mu,n}}^2 \leq \tilde{O}\left(\frac{\sqrt{(C_v^2 + dim(\boldsymbol{\theta}_\mu))N}}{1-\gamma}\right)$$

Union Bound

Lastly we use an union bound to handle the term

$$\sum_{n=1}^{N}(g_n - \nabla h_n(\theta_n))^\top \theta_N^*$$

We follow the steps in Section 9.D.3: we will use again the fact that $\theta_N^* = (\boldsymbol{\theta}_{v,N}^*, \boldsymbol{\theta}_\mu^*) \in \Theta$, so we can handle the part with $\boldsymbol{\theta}_\mu^*$ using the standard martingale concentration, and the part with $\boldsymbol{\theta}_{v,N}^*$ using the union bound.

Using the previous analyses, we see can first show that the martingale due to the part $\boldsymbol{\theta}_\mu^*$ concentrates in $\tilde{O}\left(\frac{\sqrt{Ndim(\boldsymbol{\theta}_\mu)\log(\frac{1}{\delta})}}{1-\gamma}\right)$. Likewise, using the union bound, we can show the martingale due to the part $\boldsymbol{\theta}_{v,N}^*$ concentrates in $\tilde{O}\left(\frac{\sqrt{NC_v^2\log(\frac{\mathcal{N}}{\delta})}}{1-\gamma}\right)$ where $\mathcal{N}$ some proper the covering number of the set $\left\{\boldsymbol{\theta}_v : \|\boldsymbol{\theta}_v\|_2 \leq \frac{C_v}{\sqrt{dim(\boldsymbol{\theta}_v)}}\right\}$. Because $\log\mathcal{N} = O(\dim(\boldsymbol{\theta}_v))$ for an Euclidean ball. We can combine the two bounds and show together

$$\sum_{n=1}^{N}(g_n - \nabla h_n(\theta_n))^\top \theta_N^* = \tilde{O}\left(\frac{\sqrt{N(C_v^2\dim(\boldsymbol{\theta}_v) + dim(\boldsymbol{\theta}_\mu))\log(\frac{1}{\delta})}}{1-\gamma}\right)$$

Summary

Combining the results of the three parts above, we have, with probability $1-\delta$,

$$
\begin{aligned}
&\mathrm{regret}_N(y^*_{N,\theta}) \\
&\leq \left(\sum_{n=1}^{N}(\nabla h_n(\theta_n)-g_n)^\top\theta_n\right)+\left(\max_{\theta\in\Theta}\sum_{n=1}^{N}g_n^\top(\theta_n-\theta)\right)+\left(\sum_{n=1}^{N}(g_n-\nabla h_n(\theta_n))^\top\theta^*_N\right) \\
&= \tilde{O}\left(\frac{\sqrt{N(dim(\boldsymbol{\theta_\mu})+C_v^2)\log(\frac{1}{\delta})}}{1-\gamma}\right)+\tilde{O}\left(\frac{\sqrt{(C_v^2+dim(\boldsymbol{\theta}_\mu))N}}{1-\gamma}\right) \\
&\quad+\tilde{O}\left(\frac{\sqrt{N(C_v^2\dim(\boldsymbol{\theta}_v)+dim(\boldsymbol{\theta_\mu}))\log(\frac{1}{\delta})}}{1-\gamma}\right) \\
&= \tilde{O}\left(\frac{\sqrt{Ndim(\Theta)\log(\frac{1}{\delta})}}{1-\gamma}\right)
\end{aligned}
$$

where the last step is due to C_v is a universal constant. Or equivalently, the above bounds means a sample complexity in $\tilde{O}\left(\frac{dim(\Theta)\log(\frac{1}{\delta})}{(1-\gamma)^2\epsilon^2}\right)$. Finally, we recall the policy performance has a bias $\epsilon_{\Theta,N}$ in Corollary 9.4.1 due to using function approximators. Considering this effect, we have the final statement.

CHAPTER 10

PREDICTOR-CORRECTOR POLICY OPTIMIZATION

10.1 Introduction

Reinforcement learning (RL) has recently solved a number of challenging problems (Duan et al., 2016; Mnih et al., 2013; Silver et al., 2017a). However, many of these successes are confined to games and simulated environments, where a large number of agent-environment interactions can be cheaply performed. Therefore, they are often unrealistic in real-word applications (like robotics) where data collection is an expensive and time-consuming process. Improving sample efficiency still remains a critical challenge for RL.

Model-based RL methods improve sample efficiency by leveraging an accurate model that can cheaply simulate interactions to compute policy updates in lieu of real-world interactions (Tan et al., 2018). A classical example of pure model-based methods is optimal control (Deisenroth and Rasmussen, 2011; Jacobson and Mayne, 1970; Pan and Theodorou, 2014; Todorov and Li, 2005), which has recently been extended to model abstract latent dynamics with neural networks (Oh, Singh, and Lee, 2017; Silver et al., 2017b). These methods use a (local) model of the dynamics and cost functions to predict cost-to-go functions, policy gradients, or promising improvement direction when updating policies (Anthony, Tian, and Barber, 2017; Levine and Koltun, 2013; Sun et al., 2018). Another way to use model information is the hybrid DYNA framework (Sutton, 1991; Sutton et al., 2012), which interleaves model-based and model-free updates, ideally cutting learning time in half. However, all of these approaches, while potentially accelerating policy learning, suffer from a common drawback: when the model is inaccurate, the performance of the policy can become *biased* away from the best achievable in the policy class.

Several strategies have been proposed to remove this performance bias. Learning-to-

plan attempts to train the planning process end-to-end (Amos et al., 2018; Pascanu et al., 2017; Srinivas et al., 2018), so the performance of a given planning structure is directly optimized. However, these algorithms are still optimized through standard model-free RL techniques; it is unclear as to whether they are more sample efficient. In parallel, another class of bias-free algorithms is control variate methods (Chebotar et al., 2017; Grathwohl et al., 2018; Papini et al., 2018), which use models to reduce the variance of sampled gradients to improve convergence.

In this chapter, we provide a novel learning framework that can leverage models to improve sample efficiency while avoiding performance bias due to modeling errors. Our approach is built on techniques from online learning (Gordon, 1999; Zinkevich, 2003). The use of online learning to analyze policy optimization was pioneered by Ross, Gordon, and Bagnell (2011), who proposed to reduce imitation learning (IL) to adversarial online learning problems. This reduction provides a framework for performance analysis, leading to algorithms such as DAGGER (Ross, Gordon, and Bagnell, 2011) and AGGREVATE (Ross and Bagnell, 2014). However, it was recently shown that the naïve reduction to adversarial online learning loses information (Cheng and Boots, 2018) (cf. Part I of the thesis): in practice, IL is *predictable* (Cheng et al., 2019b) and can be thought of as a predictable online learning problem (Rakhlin and Sridharan, 2012). Based on this insight, in Chapter 7 we propose a two-step algorithm, MOBIL (Cheng et al., 2019b). We prove that, by leveraging predictive models to estimate future gradients, MOBIL can speed up the convergence of IL, without incurring performance bias due to imperfect models.

Given these theoretical advances in IL, it is natural to ask if similar ideas can be extended to RL. In this chapter, we show that RL can also be formulated as a predictable online learning problem, and we propose a novel first-order learning framework, PICCOLO (PredICtor-COrrector onLine Optimization)[1], for general predictable online learning prob-

[1]In the original paper (Cheng et al., 2019d), PICCOLO was named after PredICtor-COrrector poLicy Optimization. Here we change the name to PredICtor-COrrector onLine Optimization to better emphasize the fact that PICCOLO is a generic reduction technique for online learning, not limited to policy optimization.

lems. PICCOLO is a *meta-algorithm*: it takes a standard online learning algorithm designed for adversarial problems (e.g. ADAGRAD (Duchi, Hazan, and Singer, 2011)) as input and returns a new hybrid algorithm that can use model information to accelerate convergence. This new "PICCOLOed" algorithm optimizes the policy by alternating between Prediction and Correction steps. In the Prediction Step, the learner uses a predictive model to estimate the gradient of the next loss function and then uses it to update the policy; in the Correction Step, the learner executes the updated policy in the environment, receives the true gradient , and then corrects the policy using the gradient *error*. We note that PICCOLO is orthogonal to control variate methods; it can still improve learning even in the noise-free setting (see Section 10.5.2).

Theoretically, we prove that PICCOLO can improve the convergence rate of *any* base algorithm that can be written as mirror descent (Beck and Teboulle, 2003) or Follow-the-Regularized-Leader (FTRL) (McMahan and Streeter, 2010). This family of algorithms is rich and covers most first-order algorithms used in RL and IL, as we showed in Chapter 5 (Cheng et al., 2018a). And, importantly, we show that PICCOLO does not suffer from performance bias due to model error, unlike previous model-based approaches. To validate the theory, we "PICCOLO" multiple algorithms in simulation. The experimental results show that the PICCOLOed versions consistently surpass the base algorithm and are robust to model errors.

The design of PICCOLO is made possible by a novel reduction that converts a given predictable online learning problem into a new adversarial problem, so that standard online learning algorithms can be applied optimally without referring to specialized algorithms. We show that PICCOLO includes and generalizes many existing algorithms, e.g., MOBIL, mirror-prox (Juditsky, Nemirovski, and Tauvel, 2011), and optimistic mirror descent (Rakhlin and Sridharan, 2012) (Section 10.A). Thus, we can treat PICCOLO as an automatic process for designing new algorithms that safely leverages imperfect predictive models (such as off-policy gradients or gradients simulated through dynamics models) to

speed up learning. This chapter is partly based on our paper published as (Cheng et al., 2019d).

10.2 Problem Definition

We consider solving policy optimization problems: given state and action spaces $\mathcal{S}$ and $\mathcal{A}$, and a parametric policy class Π, we desire a stationary policy $\pi \in \Pi$ that solves

$$\min_{\pi \in \Pi} \bar{J}(\pi), \quad \bar{J}(\pi) \coloneqq \mathbb{E}_{s \sim d^\pi} \mathbb{E}_{a \sim \pi | s} \left[c(s, a) \right] \tag{10.1}$$

where $c(s, a)$ is the instantaneous cost of state $s \in \mathcal{S}$ and $a \in \mathcal{A}$, $\pi(a|s)$ is the distribution of a at state s under policy π, and d^π is the average state distribution generated by running policy π in a Markov decision process (MDP); the notation $\mathbb{E}_{a \sim \pi | s}$ denotes evaluation when π is deterministic. The use of d^π in (10.1) abstracts different discrete-time RL/IL problems into a common setup (see Chapter 2). For example, an infinite-horizon γ-discounted problem with time-invariant cost c can be modeled by setting $c_t = c$ and $d^\pi(s) = \sum_{t=0}^{\infty} (1-\gamma) \gamma^t d_t^\pi(s)$, where d_t^π is the state distribution visited by policy π at time t starting from some *fixed* but unknown initial state distribution.

For convenience, we will usually omit the random variable in expectation notation (e.g. we will write (10.1) as $\mathbb{E}_{d^\pi} \mathbb{E}_\pi \left[c \right]$). For a policy π, we overload the notation π to also denote its parameter, and write Q^π and $V^\pi \coloneqq \mathbb{E}_\pi[Q^\pi]$ as its Q-function and value function, respectively.

10.3 IL and RL as Predictable Online Learning

We study policy optimization through the lens of online learning (Hazan, 2016), by treating a policy optimization algorithm as the learner in online learning and *each intermediate policy* that it produces as an online decision. This identification recasts the iterative process of policy optimization into a standard online learning setup: in round n, the learner plays

a decision $\pi_n \in \Pi$, a *per-round loss* l_n is then selected, and finally some information of l_n is revealed to the leaner for making the next decision. We note that the "rounds" considered here are the number of episodes that an algorithm interacts with the (unknown) MDP environment to obtain new information, not the time steps in the MDP. And we will suppose the learner receives an unbiased stochastic approximation $\tilde{l}_n$ of l_n as feedback.

We show that, when the per-round losses $\{l_n\}$ are properly selected, the policy performance $\{\bar{J}(\pi_n)\}$ in IL and RL can be upper bounded in terms the N-round weighted regret

$$\mathrm{regret}_N(l) \coloneqq \sum_{n=1}^{N} w_n l_n(\pi_n) - \min_{\pi\in\Pi} \sum_{n=1}^{N} w_n l_n(\pi) \tag{10.2}$$

and an expressiveness measure of the policy class Π

$$\epsilon_{\Pi,N}(l) \coloneqq \frac{1}{w_{1:N}} \min_{\pi\in\Pi} \sum_{n=1}^{N} w_n l_n(\pi) \tag{10.3}$$

where $w_n > 0$ and $w_{1:n} \coloneqq \sum_{m=1}^{n} w_m$. Moreover, we show that these online learning problems are *predictable*: that is, the per-round losses are not completely adversarial but can be estimated from past information. We will use these ideas to design PICCOLO in the next section.

10.3.1 IL as Online Learning

We start by reviewing the classical online learning approach to IL (online IL for short) (Ross, Gordon, and Bagnell, 2011) in Chapter 3 to highlight some key ideas. IL leverages domain knowledge about a policy optimization problem through expert demonstrations. Online IL, in particular, optimizes policies by letting the learner π query the expert $\pi^\star$ for desired actions, so that a policy can be quickly trained to perform as well as the expert. At its heart, online IL is based on the following lemma, which relates the performance between π and $\pi^\star$.

Lemma 10.3.1. (Kakade and Langford, 2002) *Let π and π' be two policies and $A^{\pi'}(s,a) \coloneqq Q^{\pi'}(s,a) - V^{\pi'}(s)$. Then $\bar{J}(\pi) = \bar{J}(\pi') + \mathbb{E}_{d^\pi}\mathbb{E}_\pi[A^{\pi'}]$.*

Given the equality in Lemma 10.3.1, the performance difference between π and $\pi^\star$ can then be upper-bounded as

$$\bar{J}(\pi) - \bar{J}(\pi^\star) = \mathbb{E}_{d^\pi}\mathbb{E}_\pi[A^{\pi^\star}] \leq C^{\pi^\star}\mathbb{E}_{s\sim d^\pi}[D(\pi(\cdot|s)^\star||\pi(\cdot|s))]$$

for some positive constant $C^{\pi^\star}$ and function D, which is often derived from statistical distances such as KL divergence (Cheng et al., 2018a). When A^{π^*} is available, we can also set $D(\pi(\cdot|s)^\star||\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi|s}[A^{\pi^*}(s,a)]$, as in value aggregation (AGGREVATE) (Ross and Bagnell, 2014).

Without loss of generality, let us suppose $D(\pi(\cdot|s)^\star||\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi|s}[\bar{c}(s,a)]$ for some $\bar{c}$. Online IL converts policy optimization into online learning with per-round loss

$$l_n(\pi) \coloneqq \mathbb{E}_{d^{\pi_n}}\mathbb{E}_\pi[\bar{c}]. \tag{10.4}$$

By the inequality above, it holds that $\bar{J}(\pi_n) - \bar{J}(\pi^\star) \leq C^{\pi^\star} l_n(\pi_n)$ for every n, establishing the reduction below.

Lemma 10.3.2. (Cheng et al., 2019b) *For l_n defined in* (10.4)*, $\mathbb{E}\left[\sum_{n=1}^N \frac{w_n \bar{J}(\pi_n)}{w_{1:N}}\right] \leq \bar{J}(\pi^\star) + C^{\pi^\star}\mathbb{E}\left[\epsilon_{\Pi,N}(l) + \frac{\text{regret}_N(\tilde{l})}{w_{1:N}}\right]$, where the expectation is due to sampling $\tilde{l}_n$.*

That is, when a no-regret algorithm is used, the performance concentrates toward $\bar{J}(\pi^\star) + C^{\pi^\star}\mathbb{E}[\epsilon_{\Pi,N}(l)]$.

10.3.2 RL as Online Learning

Can we also formulate RL as online learning? Here we propose a new perspective on RL using Lemma 10.3.1. Given a policy π_n in round n, we define a per-round loss

$$l_n(\pi) \coloneqq \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi}[A^{\pi_{n-1}}]. \tag{10.5}$$

which describes how well a policy π performs relative to the previous policy π_{n-1} under the state distribution of π_n. By Lemma 10.3.1, for l_n defined in (10.5), $l_n(\pi_n) = \bar{J}(\pi_n) - \bar{J}(\pi_{n-1})$ for every n, similar to the pointwise inequality of l_n that Lemma 10.3.2 is based on. With this observation, we derive the reduction below (proved in Section 10.B).

Lemma 10.3.3. *Suppose* $\frac{w_{n+k}}{w_n} \leq \frac{w_{m+k}}{w_m}$, *for all* $n \geq m \geq 1$ *and* $k \geq 0$. *For* (10.5) *and any* π_0, $\mathbb{E}[\sum_{n=1}^{N} \frac{w_n \bar{J}(\pi_n)}{w_{1:N}}] \leq \bar{J}(\pi_0) + \sum_{n=1}^{N} \frac{w_{N-n+1}}{w_{1:N}} \mathbb{E}[\mathrm{regret}_n(\tilde{l}) + w_{1:n}\epsilon_{\Pi,n}(l)]$, *where the expectation is due to sampling* $\tilde{l}_n$.

Interpretations

Lemma 10.3.3 is a policy improvement lemma, which shows that when the learning algorithm is no-regret, the policy sequence improves on-average from the initial reference policy π_0 that defines l_1. This is attributed to an important property of the definition in (10.5) that $\min_{\pi \in \Pi} l_n(\pi) \leq 0$. To see this, suppose $\mathbb{E}[\epsilon_{\Pi,n}(l)] \leq -\Omega(1)$ (i.e. there is a policy that is better than all previous n policies); this is true for small n or when the policy sequence is concentrated. Under this assumption, if $w_n = 1$ and $\mathrm{regret}_n(\tilde{l}) \leq O(\sqrt{n})$, then the average performance improves roughly $N\mathbb{E}[\epsilon_{\Pi,N}(l)]$ away from $\bar{J}(\pi_0)$.

While it is unrealistic to expect $\mathbb{E}[\epsilon_{\Pi,n}(l)] \leq 0$ for large n, we can still use Lemma 10.3.3 to comprehend *global* properties of policy improvement, for two reasons. First, the inequality in Lemma 10.3.3 holds for any interval of the policy sequence. Second, as we show in Section 10.B, the Lemma 10.3.3 also applies to dynamic regret (Zinkevich, 2003), with respect to which $\mathbb{E}[\epsilon_{\Pi,n}(l)]$ is always negative. Therefore, if an algorithm is strongly-

adaptive (Daniely, Gonen, and Shalev-Shwartz, 2015) (i.e. it is no-regret for any interval) or has sublinear dynamic regret (Jadbabaie et al., 2015), then its generated policy sequence will strictly, non-asymptotically improve. In other words, for algorithms with a stronger notion of convergence, Lemma 10.3.3 describes the global improvement rate.

Connections

The choice of per-round loss in (10.5) has an interesting relationship to both actor-critic in RL (Konda and Tsitsiklis, 2000) and AGGREVATE in IL (Ross and Bagnell, 2014).

Relationship to Actor-Critic

Although actor-critic methods, theoretically, use $\mathbb{E}_{d^{\pi_n}}(\nabla \mathbb{E}_{\pi})[A^{\pi_n}]|_{\pi=\pi_n}$ to update policy π_n, in practice, they use $\mathbb{E}_{d^{\pi_n}}(\nabla \mathbb{E}_{\pi})[A^{\pi_{n-1}}]|_{\pi=\pi_n}$, because the advantage/value function estimate in round n is updated after the policy update in order to prevent bias due to over-fitting on finite samples (Sutton and Barto, 1998). This practical gradient is *exactly* $\nabla \tilde{l}_n(\pi_n)$, the sampled gradient of (10.5). Therefore, Lemma 10.3.3 explains the properties of these practical modifications.

Relationship to Value Aggregation

AGGREVATE (Ross and Bagnell, 2014) can be viewed as taking a policy improvement step from some reference policy: e.g., with the per-round loss $\mathbb{E}_{d^{\pi_n}}\mathbb{E}_{\pi}[A^{\pi^\star}]$, it improves one step from π^*. Realizing this one step improvement in AGGREVATE, however, requires solving multiple rounds of online learning, as it effectively solves an equilibrium point problem (Cheng and Boots, 2018). Therefore, while ideally one can solve multiple AGGREVATE problems (one for each policy improvement step) to optimize policies, computationally this can be very challenging. Minimizing the loss in (10.5) can be viewed as an approximate policy improvement step in the AGGREVATE style. Rather than waiting until convergence in each AGGREVATE policy improvement step, it performs only a *single* policy update and then switches to the next AGGREVATE problem with a new reference policy (i.e. the latest policy π_{n-1}). This connection is particularly tightened if we choose $\pi_0 = \pi^\star$ and the bound

in Lemma 10.3.3 becomes relative to $\bar{J}(\pi^\star)$.

Remark 10.3.1. Based on the continuous online learning perspective in Chapter 8, it is possible to set also

$$l_n(\pi) \coloneqq \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi}[A^{\pi_n}]. \tag{10.6}$$

where the gradient $\nabla l_n(\pi_n)$ will be the policy gradient. The corresponding equilibrium problem of this per-round loss function would be finding the stationary point of the RL objective $\bar{J}(\pi)$. However, for this choice in (10.6), the policy improvement lemma, Lemma 10.3.3, would not hold.

10.3.3 Predictability

An important property of the above online learning problems is that they are not completely adversarial, as pointed out in Chapter 7 for IL (Cheng and Boots, 2018). This can be seen from the definitions of l_n in (10.4) and (10.5), respectively. For example, suppose the cost c_t in the original RL problem (10.1) is known; then the information unknown before playing the decision π_n in the environment is only the state distribution d^{π_n}. Therefore, the per-round loss cannot be truly adversarial, as the same dynamics and cost functions are used across different rounds. That is, in an idealized case where the true dynamics and cost functions are exactly known, using the policy returned from a model-based RL algorithm would incur zero regret, since only the interactions with the real MDP environment, not the model, counts as rounds. We will exploit this property to design PICCOLO.

10.4 Predictor-Corrector Learning

We showed that the performance of RL and IL can be bounded by the regret of properly constructed predictable online learning problems. These results provide a foundation for designing policy optimization algorithms: efficient learning algorithms for policy optimiza-

tion can be constructed from powerful online learning algorithms that achieve small regret. This perspective explains why common methods (e.g. mirror descent) based on gradients of (10.4) and (10.5) work well in IL and RL. However, the predictable nature of policy optimization problems suggests that directly applying these standard online learning algorithms designed for adversarial settings is *suboptimal*. The predictable information must be considered to achieve optimal convergence.

One way to include predictable information is to develop specialized two-step algorithms based on, e.g., mirror-prox or FTRL-prediction (Ho-Nguyen and Kılınç-Karzan, 2018; Juditsky, Nemirovski, and Tauvel, 2011; Rakhlin and Sridharan, 2012). For IL, MoBIL was recently proposed (Cheng et al., 2019b) (cf. Chapter 7), which updates policies by approximate Be-the-Leader (Kalai and Vempala, 2005) and provably achieves faster convergence than previous methods. However, these two-step algorithms often have obscure and non-sequential update rules, and their adaptive and accelerated versions are less accessible (Diakonikolas and Orecchia, 2017). This can make it difficult to implement and tune them in practice.

Here we take an alternative, *reduction-based* approach. We present PICCOLO, a general first-order framework for solving predictable online learning problems. PICCOLO is a meta-algorithm that turns a base algorithm designed for adversarial problems into a new algorithm that can leverage the predictable information to achieve better performance. As a result, we can adopt sophisticated first-order adaptive algorithms to optimally learn policies, without reinventing the wheel. Specifically, given *any* first-order base algorithm belonging to the family of (adaptive) mirror descent and FTRL algorithms, we show how one can "PICCOLO it" to achieve a faster convergence rate without introducing additional performance bias due to prediction errors. Most first-order policy optimization algorithms belong to this family (see Chapter 5), so we can PICCOLO these model-free algorithms into new hybrid algorithms that can robustly use (imperfect) predictive models, such as off-policy gradients and simulated gradients, to improve policy learning.

10.4.1 The PICCOLO Idea

The design of PICCOLO is based on the observation that an N-round predictable online learning problem can be written as a new adversarial problems with $2N$ rounds. To see this, let $\{l_n\}_{n=1}^N$ be the original predictable loss sequence. Suppose, before observing l_n, we have access to a *model loss* $\hat{l}_n$ that contains the predictable information of l_n. Define $\delta_n = l_n - \hat{l}_n$. We can then write the accumulated loss (which regret concerns) as $\sum_{n=1}^N l_n(\pi_n) = \sum_{n=1}^N \hat{l}_n(\pi_n) + \delta_n(\pi_n)$. That is, we can view the predictable problem with $\{l_n\}_{n=1}^N$ as a new adversarial online learning problem with a loss sequence $\hat{l}_1, \delta_1, \hat{l}_2, \delta_2, \ldots, \hat{l}_N, \delta_N$.

The idea of PICCOLO is to apply standard online learning algorithms designed for adversarial settings to this new $2N$-round problem. This would create a new set of decision variables $\{\hat{\pi}_n\}_{n=1}^N$, in which $\hat{\pi}_n$ denotes the decision made before seeing $\hat{l}_n$, and leads to the following sequence $\pi_1, \delta_1, \hat{\pi}_2, \hat{l}_2, \pi_2, \delta_2, \ldots$ (in which we define $\delta_1 = l_1$). We show that when the base algorithm is optimal in adversarial settings, this simple strategy results in a decision sequence $\{\pi_n\}_{n=1}^N$ whose regret with respect to $\{l_n\}_{n=1}^N$ is optimal, just as those specialized two-step algorithms (Ho-Nguyen and Kılınç-Karzan, 2018; Juditsky, Nemirovski, and Tauvel, 2011; Rakhlin and Sridharan, 2012). In Section 10.A, we show PICCOLO unifies and generalize these two-step algorithms to be adaptive.

10.4.2 The Meta Algorithm PICCOLO

We provide details to realize this reduction. We suppose, in round n, the model loss is given as $\hat{l}_n(\pi) = \langle \hat{g}_n, \pi \rangle$ for some vector $\hat{g}_n$, and stochastic first-order feedback $g_n = \nabla \tilde{l}_n(\pi_n)$ from l_n is received. Though this linear form of model loss seems restrictive, later in Section 10.4.2 we will show that it is sufficient to represent predictable information.

Base Algorithms

We first give a single description of different base algorithms for the formal definition of the reduction steps. Here we limit our discussions to mirror descent and postpone the

FTRL case to Section 10.C. We assume that Π is a convex compact subset in some normed space with norm $\|\cdot\|$, and we use $B_R(\pi||\pi') = R(\pi) - R(\pi') - \langle \nabla R(\pi'), \pi - \pi' \rangle$ to denote a Bregman divergence generated by a strictly convex function R, called the distance generator.

Mirror descent updates decisions based on proximal maps. In round n, given direction g_n and weight w_n, it executes

$$\pi_{n+1} = \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + B_{R_n}(\pi||\pi_n) \tag{10.7}$$

where R_n is a strongly convex function; (10.7) reduces to gradient descent with step size η_n when $R_n(\cdot) = \frac{1}{2\eta_n}\|\cdot\|^2$. More precisely, (10.7) is composed of two steps: 1) the update of the distance generator to R_n, and 2) the update of the decision to π_{n+1}; different mirror descent algorithms differ in how the regularization is selected and adapted.

PICCOLO explicitly treats a base algorithm as the composition of two basic operations (this applies also to FTRL)

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, H_{n-1}, g_n, w_n) \\ h_{n+1} &= \texttt{update}(h_n, H_n, g_n, w_n) \end{aligned} \tag{10.8}$$

so that later it can recompose them to generate the new algorithm. For generality, we use h and H to denote the abstract representations of the decision variable and the regularization, respectively. In mirror descent, h is exactly the decision variable, H is the distance generator, and we can write $\texttt{update}(h, H, g, w) = \arg\min_{\pi' \in \Pi} \langle wg, \pi' \rangle + B_H(\pi'||h)$. The operation `adapt` denotes the algorithm-specific scheme for the regularization update (e.g. changing the step size), which in general updates the size of regularization to grow slowly and inversely proportional to the norm of g_n.

The PICCOLO*ed Algorithm*

PICCOLO generates decisions by applying a given base algorithm in (10.8) to the new problem with losses $\delta_1, \hat{l}_2, \delta_2, \ldots$. This is accomplished by recomposing the basic operations in (10.8) into the Prediction and the Correction Steps:

$$h_n = \texttt{update}(\hat{h}_n, H_{n-1}, \hat{g}_n, w_n) \qquad \text{[Prediction]}$$

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, H_{n-1}, e_n, w_n) \\ \hat{h}_{n+1} &= \texttt{update}(h_n, H_n, e_n, w_n) \end{aligned} \qquad \text{[Correction]}$$

where $\hat{h}_n$ is the abstract representation of $\hat{\pi}_n$, and $e_n = g_n - \hat{g}_n$ is the error direction. We can see that the Prediction and Correction Steps are exactly the update rules resulting from applying (10.8) to the new adversarial problem, except that only h_n is updated in the Prediction Step, *not* the regularization (i.e. the step size). This asymmetry design is important for achieving optimal regret, because in the end we care only about the regret of $\{\pi_n\}$ on the original loss sequence $\{l_n\}$.

In round n, the "PICCOLOed" algorithm first performs the Prediction Step using $\hat{g}_n$ to generate the learner's decision (i.e. π_n) and runs this new policy in the environment to get the true gradient g_n. Using this feedback, the algorithm performs the Correction Step to amend the bias of using $\hat{g}_n$. This is done by first adapting the regularization to H_n and then updating π_n to $\hat{\pi}_{n+1}$ along the error $e_n = g_n - \hat{g}_n$.

Model Losses and Predictive Models

The Prediction Step of PICCOLO relies on the vector $\hat{g}_n$ to approximate the future gradient g_n. Here we discuss different ways to specify $\hat{g}_n$ based on the concept of predictive models in Chapter 7. A *predictive model* Φ_n is a first-order oracle such that $\Phi_n(\cdot)$ approximates $\nabla l_n(\cdot)$. In practice, a predictive model can be a simulator with an (online learned) dynamics model (Deisenroth and Rasmussen, 2011; Tan et al., 2018), or a neural network trained

to predict the required gradients (Oh, Singh, and Lee, 2017; Silver et al., 2017b). An even simpler heuristic is to construct predictive models by *off-policy* gradients $\Phi_n(\cdot) = \sum_{m=n-K}^{n-1} \nabla \tilde{l}_m(\cdot)$ where K is the buffer size.

In general, we wish to set $\hat{g}_n$ to be close to g_n, as we will later show in Section 10.5 that the convergence rate of PICCOLO depends on their distance. However, even when we have perfect predictive models, this is still a non-trivial task. We face a chicken-or-the-egg problem: g_n depends on π_n, which in turn depends on $\hat{g}_n$ from the Prediction Step.

In Chapter 7, we show one effective heuristic is to set $\hat{g}_n = \Phi_n(\hat{\pi}_n)$, because we may treat $\hat{\pi}_n$ as an estimate of π_n. However, due to the mismatch between $\hat{\pi}_n$ and π_n, this simple approach has errors even when the predictive model is perfect. To better leverage a given predictive model, we propose to solve for $\hat{g}_n$ and π_n *simultaneously*. That is, we wish to solve a fixed-point problem, finding h_n such that

$$h_n = \texttt{update}(\hat{h}_n, H_{n-1}, \Phi_n(\pi_n(h_n)), w_n) \tag{10.9}$$

The exact formulation of the fixed-point problem depends on the class of base algorithms. For mirror descent, it is a variational inequality: find $\pi_n \in \Pi$ such that $\forall \pi \in \Pi$,

$$\langle \Phi_n(\pi_n) + \nabla R_{n-1}(\pi_n) - \nabla R_{n-1}(\hat{\pi}_n), \pi - \pi_n \rangle \geq 0.$$

In a special case when $\Phi_n = \nabla f_n$ for some function f_n, the above variational inequality is equivalent to finding a stationary point of the optimization problem $\min_{\pi \in \Pi} f_n(\pi) + B_{R_{n-1}}(\pi || \hat{\pi}_n)$. In other words, one way to implement the Prediction Step is to solve the above minimization problem for π_n and use $\nabla f_n(\pi_n)$ as the effective prediction $\hat{g}_n$.

10.4.3 Summary: Why Does PICCOLO Work?

We provide a summary of the full algorithm for policy optimization in Algorithm 4. We see that PICCOLO uses the predicted gradient to take an extra step to accelerate learning,

and, meanwhile, to prevent the error accumulation, it adaptively adjusts the step size (i.e. the regularization) based on the prediction error and corrects for the bias on the policy right away. To gain some intuition, let us consider ADAGRAD (Duchi, Hazan, and Singer, 2011) as a base algorithm[2]:

$$\begin{aligned} G_n &= G_{n-1} + \mathrm{diag}(w_n g_n \odot w_n g_n) \\ \pi_{n+1} &= \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + \frac{1}{2\eta}(\pi - \pi_n)^\top G_n^{1/2}(\pi - \pi_n) \end{aligned}$$

where $G_0 = \epsilon I$ and $\eta, \epsilon > 0$, and $\odot$ denotes element-wise multiplication. This update has an `adapt` operation as $\texttt{adapt}(h, H, g, w) = G + \mathrm{diag}(wg \odot wg)$ which updates the Bregman divergence based on the gradient size.

PICCOLO transforms ADAGRAD into a new algorithm. In the Prediction Step, it performs

$$\pi_n = \arg\min_{\pi \in \Pi} \langle w_n \hat{g}_n, \pi \rangle + \frac{1}{2\eta}(\pi - \pi_{n-1})^\top G_{n-1}^{1/2}(\pi - \pi_{n-1})$$

In the Correction Step, it performs

$$\begin{aligned} G_n &= G_{n-1} + \mathrm{diag}(w_n e_n \odot w_n e_n) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi \in \Pi} \langle w_n e_n, \pi \rangle + \frac{1}{2\eta}(\pi - \hat{\pi}_n)^\top G_n^{1/2}(\pi - \hat{\pi}_n) \end{aligned}$$

We see that the PICCOLO-ADAGRAD updates G_n proportional to the prediction error e_n instead of g_n. It takes larger steps when models are accurate, and decreases the step size once the prediction deviates. As a result, PICCOLO is robust to model quality: it accelerates learning when the model is informative, and prevents inaccurate (potentially adversarial) models from hurting the policy. We will further demonstrate this in theory and in the experiments.

[2]We provide another example in Section 10.E.

[‡]Here we assume `project` is automatically performed inside `PredictionStep` and `CorrectionStep`.

Algorithm 4 PICCOLO

Input: policy π_1, regularization H_0, model Φ_1, iteration N, exponent p
Output: $\bar{\pi}_N$

Set $\hat{\pi}_1 = \pi_1$ and weights $w_n = n^p$
Sample integer $K \in [1, N]$ with $P(K = n) \propto w_n$
for $n = 1 \dots K - 1^{\ddagger}$ **do**
    $\pi_n, \hat{g}_n = \texttt{PredictionStep}(\hat{\pi}_n, \Phi_n, H_{n-1}, w_n)$
    $\mathcal{D}_n, g_n = \texttt{DataCollection}(\pi_n)$
    $H_n, \hat{\pi}_{n+1} = \texttt{CorrectionStep}(\pi_n, e_n, H_{n-1}, w_n)$, where $e_n = g_n - \hat{g}_n$.
    $\Phi_{n+1} = \texttt{ModelUpdate}(\Phi_n, \mathcal{D})$, where $\mathcal{D} = \mathcal{D} \bigcup \mathcal{D}_n$.
end for
Set $\bar{\pi}_N = \pi_{K-1}$

10.5 Theoretical Analysis

In this section, we show that PICCOLO has two major benefits over previous approaches: 1) it accelerates policy learning when the models predict the required gradient well on average; and 2) it does not bias the performance of the policy, even when the prediction is incorrect.

To analyze PICCOLO, we introduce an assumption to quantify the `adapt` operator of a base algorithm.

Assumption 10.5.1. `adapt` chooses a regularization sequence such that, for some $M_N = o(w_{1:N})$, $\|H_0\|_{\mathcal{R}} + \sum_{n=1}^{N} \|H_n - H_{n-1}\|_{\mathcal{R}} \leq M_N$ for some norm $\|\cdot\|_{\mathcal{R}}$ which measures the size of regularization.

This assumption, which requires the regularization to increase slower than the growth of $w_{1:N}$, is satisfied by most reasonably-designed base algorithms. For example, in a uniformly weighted problem, gradient descent with a decaying step size $O(\frac{1}{\sqrt{n}})$ has $M_N = O(\sqrt{N})$. In general, for stochastic problems, an optimal base algorithm would ensure $M_N = O(\frac{w_{1:N}}{\sqrt{N}})$.

10.5.1 Convergence Properties

Now we state the main result, which quantifies the regret of PICCOLO with respect to the sequence of linear loss functions that it has access to. The proof is given in Section 10.F.

Theorem 10.5.1. *Suppose H_n defines a strongly convex function with respect to $\|\cdot\|_n$. Under Assumption 10.5.1, running* PICCOLO *ensures $\sum_{n=1}^N \langle w_n g_n, \pi_n - \pi\rangle \le M_N + \sum_{n=1}^N \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$, for all $\pi \in \Pi$.*

The term $\|e_n\|_{*,n}^2$ in Theorem 10.5.1 says that the performance of PICCOLO depends on how well the base algorithm adapts to the error e_n through the `adapt` operation in the Correction Step. Usually `adapt` updates H_n gradually (Assumption 10.5.1) while minimizing $\frac{1}{2}\|e_n\|_{*,n}^2$, like we showed in ADAGRAD.

In general, when the base algorithm is adaptive and optimal for adversarial problems, we show in Section 10.G that its PICCOLOed version guarantees that, for any π,

$$\mathbb{E}[\sum_{n=1}^N \langle w_n g_n, \pi_n - \pi\rangle] \le O(1) + C_{\Pi,\Phi}\frac{w_{1:N}}{\sqrt{N}},$$

where $C_{\Pi,\Phi} = O(|\Pi| + E_\Phi + \sigma_g^2 + \sigma_{\hat{g}}^2)$ is some constant related to the diameter of Π (denoted as $|\Pi|$), the model bias E_Φ, and the sampling variance σ_g^2 and $\sigma_{\hat{g}}^2$ of g_n and $\hat{g}_n$, respectively. Through Lemma 10.3.2 and 10.3.3, this bound directly implies accelerated and bias-free policy performance.

Theorem 10.5.2. *Suppose $\tilde{l}_n$ is convex[4] and $w_n \ge \Omega(1)$. Then running* PICCOLO *yields $\mathbb{E}[\mathrm{regret}_N(\tilde{l})/w_{1:N}] = O(\frac{C_{\Pi,\Phi}}{\sqrt{N}})$, where $C_{\Pi,\Phi} = O(|\Pi| + E_\Phi + \sigma_g^2 + \sigma_{\hat{g}}^2) = O(1)$.*

Table 10.1: Upper bounds of the average regret of different policy optimization algorithms.

ALGORITHMS	UPPER BOUNDS IN BIG-O
PICCOLO	$\frac{1}{\sqrt{N}}\left(\lvert\Pi\rvert + \sigma_g^2 + \sigma_{\hat{g}}^2 + E_\Phi\right)$
MODEL-FREE	$\frac{1}{\sqrt{N}}\left(\lvert\Pi\rvert + G_g^2 + \sigma_g^2\right)$
MODEL-BASED	$\frac{1}{\sqrt{N}}\left(\lvert\Pi\rvert + G_{\hat{g}}^2 + \sigma_{\hat{g}}^2\right) + E_\Phi$
DYNA	$\frac{1}{\sqrt{2N}}\left(\lvert\Pi\rvert + \frac{1}{2}\left(G_g^2 + G_{\hat{g}}^2 + \sigma_g^2 + \sigma_{\hat{g}}^2\right)\right) + E_\Phi$

10.5.2 Comparison

To appreciate the advantages of PICCOLO, we review several policy optimization algorithms and compare their regret. We show that they can be viewed as incomplete versions of PICCOLO, which only either result in accelerated learning *or* are unbiased, but not both (see in Table 10.1).

We first consider the common model-free approach (Cheng et al., 2018a; Kakade, 2002; Peters, Mülling, and Altun, 2010; Peters and Schaal, 2008; Silver et al., 2014; Sun et al., 2017; Sutton et al., 2000), i.e. applying the base algorithm with g_n. To make the comparison concrete, suppose $\|\mathbb{E}[g_n]\|_*^2 \leq G_g^2$ for some constant G_g, where we recall g_n is the sampled true gradient. As the model-free approach is equivalent to setting $\hat{g}_n = 0$ in PICCOLO, by Theorem 10.5.1 (with $e_n = g_n$), the constant C_Π in Theorem 10.5.2 would become $O(|\Pi| + G_g^2 + \sigma_g^2)$. In other words, PICCOLOing the base algorithm improves the constant factor from G_g^2 to $\sigma_{\hat{g}}^2 + E_\Phi$. Therefore, while the model-free approach is bias-free, its convergence can be further improved by PICCOLO, as long as the models $\{\Phi_n\}$ are reasonably accurate on average.[5]

Next we consider the pure model-based approach with a model that is potentially learned online (Deisenroth and Rasmussen, 2011; Jacobson and Mayne, 1970; Levine and

[4]The convexity assumption is standard, as used in (Cheng and Boots, 2018; Duchi, Hazan, and Singer, 2011; Kingma and Ba, 2014; Ross, Gordon, and Bagnell, 2011), which holds for tabular problems as well as some special cases, like continuous-time problems (cf. (Cheng and Boots, 2018)).

[5]It can be shown that if the model is learned online with a no-regret algorithm, it would perform similarly to the best model in the hindsight (cf. Section 10.G.4)

Koltun, 2013; Pan and Theodorou, 2014; Sun et al., 2018; Todorov and Li, 2005). As this approach is equivalent to only performing the Prediction Step[6], its performance suffers from any modeling error. Specifically, suppose $\|\mathbb{E}[\hat{g}_n]\|_*^2 \leq G_{\hat{g}}^2$ for some constant $G_{\hat{g}}$. One can show that the bound in Theorem 10.5.2 would become $O((|\Pi| + G_{\hat{g}}^2 + \sigma_{\hat{g}}^2)/\sqrt{N} + E_\Phi)$, introducing a constant bias in $O(E_\Phi)$.

A hybrid heuristic to combine the model-based and model-free updates is DYNA (Sutton, 1991; Sutton et al., 2012), which interleaves the two steps during policy optimization. This is equivalent to applying g_n, instead of the error e_n, in the Correction Step of PICCOLO. Following a similar analysis as above, one can show that the convergence rate in Theorem 10.5.2 would become $O((|\Pi| + G^2 + \sigma^2)/\sqrt{2N} + E_\Phi)$, where $G^2 = \frac{1}{2}(G_g^2 + G_{\hat{g}}^2)$ and $\sigma^2 = \frac{1}{2}(\sigma_g^2 + \sigma_{\hat{g}}^2)$. Therefore, DYNA is effectively twice as fast as the pure model-free approach when the model is accurate. However, it would eventually suffer from the performance bias due model error, as reflected in the term E_Φ. We will demonstrate this property experimentally in Figure 10.1.

Finally, we note that the idea of using Φ_n as control variate (Chebotar et al., 2017; Grathwohl et al., 2018; Papini et al., 2018) is orthogonal to the setups considered above, and it can be naturally combined with PICCOLO. For example, we can also use Φ_n to compute a better sampled gradient g_n with smaller variance (line 5 of Algorithm 4). This would improve σ_g^2 in the bounds of PICCOLO to a smaller $\tilde{\sigma}_g^2$, the size of reduced variance.

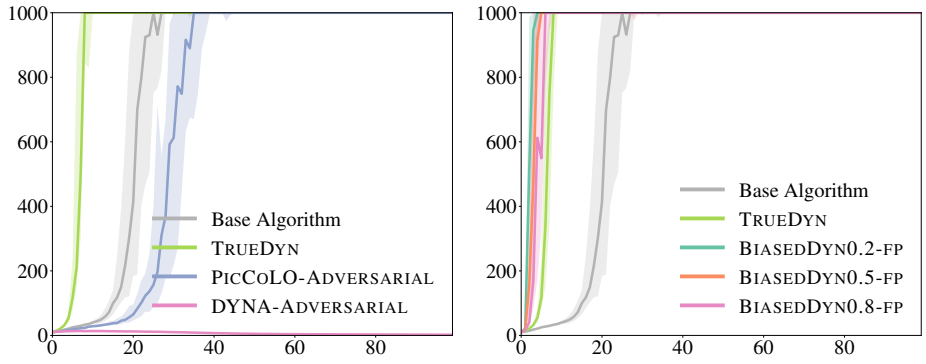
10.6 Experiments

We corroborate our theoretical findings with experiments[7] in learning neural network policies to solve robot RL tasks (CartPole, Hopper, Snake, and Walker3D) from OpenAI Gym (Brockman et al., 2016) with the DART physics engine (Lee et al., 2018a)[8]. The aim is to see if PICCOLO improves the performance of a base algorithm, even though in

[6]These algorithms can be realized by the fixed-point formulation of the Prediction Step with (arbitrarily small) regularization.

[7]The codes are available at `https://github.com/gtrll/rlfamily`.

[8]The environments are defined in DartEnv, hosted at `https://github.com/DartEnv`.



(a) Adversarial model (b) Different model fidelities

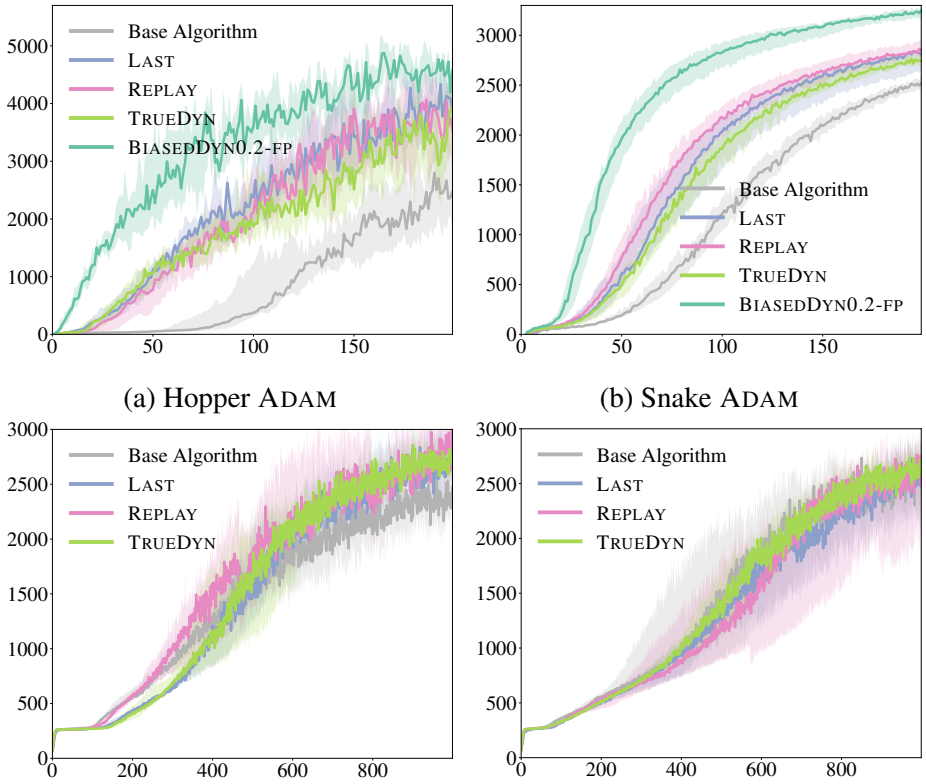
Figure 10.1: Performance of PICCOLO with different predictive models. x axis is iteration number and y axis is sum of rewards. The curves are the median among 8 runs with different seeds, and the shaded regions account for 25% percentile. ADAM is used as the base algorithm, and the update rule, by default, is PICCOLO; e.g. TRUEDYN in (a) refers to PICCOLO with TRUEDYN predictive model. (a) Comparison of PICCOLO and DYNA with adversarial model. (b) PICCOLO with the fixed-point setting (10.9) with dynamics model in different fidelities. BIASEDDYN0.8 indicates that the mass of each individual robot link is either increased or decreased by 80% with probability 0.5 respectively.

these experiments the convexity assumption in the theory does not hold. We choose several popular first-order mirror descent base algorithms (ADAM (Kingma and Ba, 2014), natural gradient descent NATGRAD (Kakade, 2002), and trust-region optimizer TRPO (Schulman et al., 2015b)). We compute g_n by GAE (Schulman et al., 2015a). For predictive models, we consider off-policy gradients (with the samples of the last iteration LAST or a replay buffer REPLAY) and gradients computed through simulations with the true or biased dynamics models (TRUEDYN or BIASEDDYN). We will label a model with FP if $\hat{g}_n$ is determined by the fixed-point formulation (10.9)[9]; otherwise, $\hat{g}_n = \Phi_n(\hat{\pi}_n)$. Please refer to Section 10.H for the details.

In Fig. 10.1, we first use CartPole to study Theorem 10.5.2, which suggests that PICCOLO is unbiased and improves the performance when the prediction is accurate. Here we additionally consider an extremely bad model, ADVERSARIAL, that predicts the gradients adversarially.[10] Figure 10.1 (a) illustrates the performance of PICCOLO and DYNA, when

[9]In implementation, we solve the corresponding optimization problem with a few number of iterations. For example, BIASEDDYN-FP is aporoximatedly solved with 5 iterations.

[10]We set $\hat{g}_{n+1} = -\left(\max_{m=1,\dots,n} \|g_m\|/\|g_n\|\right) g_n$.



(a) Hopper ADAM (b) Snake ADAM

(c) Walker3D NATGRAD (d) Walker3D TRPO

Figure 10.2: Performance of PICCOLO in various tasks. x axis is iteration number and y axis is sum of rewards. The curves are the median among 8 runs with different seeds, and the shaded regions account for 25% percentile.

ADAM is chosen as the base algorithm. We observe that PICCOLO improves the performance when the model is accurate (i.e. TRUEDYN). Moreover, PICCOLO is robust to modeling errors. It still converges when the model is adversarially attacking the algorithm, whereas DYNA fails completely. In Fig. 10.1 (b), we conduct a finer comparison of the effects of different model accuracies (BIASEDDYN-FP), when $\hat{g}_n$ is computed using (10.9). To realize inaccurate dynamics models to be used in the Prediction step, we change the mass of links of the robot by a certain factor, e.g. BIASEDDYN0.8 indicates that the mass of each individual link is either increased or decreased by 80% with probability 0.5, respectively. We see that the fixed-point formulation (10.9), which makes multiple queries of Φ_n for computing $\hat{g}_n$, performs much better than the heuristic of setting $\hat{g}_n = \Phi(\hat{\pi}_n)$, even when the latter is using the true model (TRUEDYN). Overall, we see PICCOLO with BIASEDDYN-FP is able to accelerate learning, though with a degree varying with model accuracies; but even for models with a large bias, it still converges unbiasedly, as we previously observed in Fig. 10.1 (a),

In Fig. 10.2, we study the performance of PICCOLO in a range of environments. In general, we find that PICCOLO indeed improves the performance[11] though the exact degree depends on how $\hat{g}_n$ is computed. In Fig. 10.2 (a) and (b), we show the results of using ADAM as the base algorithm. We observe that, while setting $\hat{g}_n = \Phi_n(\hat{\pi}_n)$ is already an effective heuristic, the performance of PICCOLO can be further and largely improved if we adopt the fixed-point strategy in (10.9), as the latter allows the learner to take more globally informed update directions. Finally, to demonstrate the flexibility of the proposed framework, we also "PICCOLO" two other base algorithms, NATGRAD and TRPO, in Fig. 10.2 (c) and (d), respectively. The complete set of experimental results can be found in Section 10.H.

[11]Note that different base algorithms are not directly comparable, as further fine-tuning of step sizes is required.

10.7 Conclusion

PICCOLO is a general reduction-based framework for solving predictable online learning problems. It can be viewed as an automatic strategy for generating new algorithms that can leverage prediction to accelerate convergence. Furthermore, PICCOLO uses the Correction Step to recover from the mistake made in the Prediction Step, so the presence of modeling errors does not bias convergence, as we show in both the theory and experiments. The design of PICCOLO leaves open the question of how to design good predictive models. While PICCOLO is robust against modeling error, the accuracy of a predictive model can affect its effectiveness. PICCOLO only improves the performance when the model can make non-trivial predictions. In the experiments, we found that off-policy and simulated gradients are often useful, but they are not perfect. It would be interesting to see whether a predictive model that is trained to directly minimize the prediction error can further help policy learning. Finally, we note that, despite the focus of this chapter on policy optimization, PICCOLO can naturally be applied to other optimization and learning problems.

10.A Relationship between PICCOLO and Existing Algorithms

We discuss how the framework of PICCOLO unifies existing online learning algorithms and provides their natural adaptive generalization. To make the presentation clear, we summarize the effective update rule of PICCOLO when the base algorithm is mirror descent

$$\begin{aligned} \pi_n &= \arg\min_{\pi \in \Pi} \langle w_n \hat{g}_n, \pi \rangle + B_{R_{n-1}}(\pi || \hat{\pi}_n) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi \in \Pi} \langle w_n e_n, \pi \rangle + B_{R_n}(\pi || \pi_n) \end{aligned} \tag{10.10}$$

and that when the base algorithm is FTRL,

$$\pi_n = \arg\min_{\pi \in \Pi} \langle w_n \hat{g}_n, \pi \rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi || \pi_m)$$
$$\hat{\pi}_{n+1} = \arg\min_{\pi \in \Pi} \langle w_n e_n, \pi \rangle + B_{r_n}(\pi || \pi_n) + \langle w_n \hat{g}_n, \pi \rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi || \pi_m) \tag{10.11}$$

Because $e_n = g_n - \hat{g}_n$, PICCOLO with FTRL exactly matches the update rule (MOBIL) proposed by Cheng et al. (2018a)

$$\begin{aligned} \pi_n &= \arg\min_{\pi \in \Pi} \langle w_n \hat{g}_n, \pi \rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi || \pi_m) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi || \pi_m) \end{aligned} \tag{10.12}$$

As comparisons, we consider existing two-step update rules, which in our notation can be written as follows:

- Extragradient descent (Korpelevich, 1976), mirror-prox (Juditsky, Nemirovski, and Tauvel, 2011; Nemirovski, 2004) or optimistic mirror descent (Chiang et al., 2012; Rakhlin and Sridharan, 2012)

$$\begin{aligned} \pi_n &= \arg\min_{\pi \in \Pi} \langle \hat{g}_n, \pi \rangle + B_R(\pi || \hat{\pi}_n) \\ \hat{\pi}_{n+1} &= \arg\min_{\pi \in \Pi} \langle g_n, \pi \rangle + B_R(\pi || \hat{\pi}_n) \end{aligned} \tag{10.13}$$

- FTRL-with-Prediction/optimistic FTRL (Rakhlin and Sridharan, 2012)

$$\pi_n = \arg\min_{\pi \in \Pi} R(\pi) + \langle \hat{g}_n, \pi \rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi \rangle \tag{10.14}$$

Let us first review the previous update rules. Originally extragradient descent (Korpelevich, 1976) and mirror prox (Juditsky, Nemirovski, and Tauvel, 2011; Nemirovski,

2004) were proposed to solve VIs (the latter is an extension to consider general Bregman divergences). As pointed out by Cheng et al. (2019b), when applied to an online learning problem, these algorithms effectively assign $\hat{g}_n$ to be the online gradient as if the learner plays a decision at $\hat{\pi}_n$. On the other hand, in the online learning literature, optimistic mirror descent (Chiang et al., 2012) was proposed to use $\hat{g}_n = g_{n-1}$. Later Rakhlin and Sridharan, 2012 generalized it to use some arbitrary sequence $\hat{g}_n$, and provided a FTRL version update rule in (10.14). However, it is unclear in (Rakhlin and Sridharan, 2012) where the prediction $\hat{g}_n$ comes from in general, though they provide an example in the form of learning from experts.

Recently Cheng et al. (2018a) generalized the FTRL version of these ideas to design MOBIL, which introduces extra features 1) use of weights 2) non-stationary Bregman divergences (i.e. step size) and 3) the concept of predictive models ($\Phi_n \approx \nabla l_n$). The former two features are important to speed up the convergence rate of IL. With predictive models, they propose a conceptual idea (inspired by Be-the-Leader) which solves for π_n by the VI of finding π_n such that

$$\left\langle w_n \Phi_n(\pi_n) + \sum_{m=1}^{n} w_m g_m, \pi' - \pi_n \right\rangle \geq 0 \qquad \forall \pi' \in \Pi \tag{10.15}$$

and a more practical version (10.12) which sets $\hat{g}_n = \Phi_n(\pi_n)$. Under proper assumptions, they prove that the practical version achieves the same rate of non-asymptotic convergence as the conceptual one, up to constant factors.

PICCOLO unifies and generalizes the above update rules. We first notice that when the weight is constant, the set Π is unconstrained, and the Bregman divergence is constant, PICCOLO with mirror descent in (10.10) is the same as (10.13), i.e.,

$$\begin{aligned}
\hat{\pi}_{n+1} &= \arg\min_{\pi \in \Pi} \langle e_n, \pi \rangle + B_R(\pi || \pi_n) \\
&= \arg\min_{\pi \in \Pi} \langle e_n, \pi \rangle + R(\pi) - \langle \nabla R(\pi_n), \pi \rangle
\end{aligned}$$

$$
\begin{aligned}
&= \arg\min_{\pi\in\Pi} \langle g_n - \hat{g}_n, \pi\rangle + R(\pi) - \langle \nabla R(\hat{\pi}_n) - \hat{g}_n, \pi\rangle \\
&= \arg\min_{\pi\in\Pi} \langle g_n, \pi\rangle + R(\pi) - \langle \nabla R(\hat{\pi}_n), \pi\rangle \\
&= \arg\min_{\pi\in\Pi} \langle g_n, \pi\rangle + B_R(\pi||\hat{\pi}_n)
\end{aligned}
$$

Therefore, PICCOLO with mirror descent includes previous two-step algorithms with proper choices of $\hat{g}_n$. On the other hand, we showed above that PICCOLO with FTRL (10.11) recovers exactly (10.12).

PICCOLO further generalizes these updates in two important aspects. First, it provides a systematic way to make these mirror descent and FTRL algorithms *adaptive*, by the reduction that allows reusing existing adaptive algorithm designed for adversarial settings. By contrast, all the previous update schemes discussed above (even MOBIL) are based on constant or pre-scheduled Bregman divergences, which requires the knowledge of several constants of problem properties that are usually unknown in practice. The use of adaptive schemes more amenable to hyperparameter tuning in practice. We note that it is possible to make mirror-prox-like algorithms adaptive too, which leads to update rule in the form

$$
\begin{aligned}
\pi_n &= \arg\min_{\pi\in\Pi} \langle \hat{g}_n, \pi\rangle + B_{R_{n-1}}(\pi||\hat{\pi}_n) \\
\hat{\pi}_{n+1} &= \arg\min_{\pi\in\Pi} \langle g_n, \pi\rangle + B_{R_{n-1}}(\pi||\hat{\pi}_n)
\end{aligned}
\tag{10.16}
$$

Notice when updating along direction of g_n in the above law, the regularization R_{n-1} is one-step delayed compared with PICCOLO which uses R_n that includes the information of g_n in updating to $\hat{\pi}_{n+1}$. This delay is because these two steps in mirror-prox-like algorithms need to couple the same distance generating function in defining the Bregman divergences.

Second, PICCOLO generalize the use of predictive models from the VI formulation in (10.15) to the *fixed-point* formulation in (10.9). One can show that when the base algorithm is FTRL and we remove the Bregman divergence[12], (10.9) is the same as (10.15).

[12]Originally the conceptual MOBIL algorithm is based on the assumption that l_n is strongly convex and

In other words, (10.15) essentially can be viewed as a mechanism to find $\hat{g}_n$ for (10.12). But importantly, the fixed-point formulation is method agnostic and therefore applies to also the mirror descent case. In particular, in Section 10.4.2, we point out that when Φ_n is a gradient map, the fixed-point problem reduces to finding a stationary point[13] of a non-convex optimization problem. This observation makes implementation of the fixed-point idea much easier and more stable in practice (as we only require the function associated with Φ_n to be lower bounded to yield a stable problem).

10.B Proof of Lemma 10.3.3

Without loss of generality we suppose $w_1 = 1$ and $\bar{J}(\pi) \geq 0$ for all π. And we assume the weighting sequence $\{w_n\}$ satisfies, for all $n \geq m \geq 1$ and $k \geq 0$, $\frac{w_{n+k}}{w_n} \leq \frac{w_{m+k}}{w_m}$. This means $\{w_n\}$ is an non-decreasing sequence and it does not grow faster than exponential (for which $\frac{w_{n+k}}{w_n} = \frac{w_{m+k}}{w_m}$). For example, if $w_n = n^p$ with $p \geq 0$, it easy to see that

$$\frac{(n+k)^p}{n^p} \leq \frac{(m+k)^p}{m^p} \impliedby \frac{n+k}{n} \leq \frac{m+k}{m} \impliedby \frac{k}{n} \leq \frac{k}{m}$$

For simplicity, let us first consider the case where l_n is deterministic. Given this assumption, we bound the performance in terms of the weighted regret below. For l_n defined in (10.5), we can write

$$\begin{aligned}
&\sum_{n=1}^{N} w_n \bar{J}(\pi_n) \\
&= \sum_{n=1}^{N} w_n \bar{J}(\pi_{n-1}) + w_n \mathbb{E}_{d^{\pi_n}} \mathbb{E}_{\pi_n}[A^{\pi_{n-1}}] \\
&= \sum_{n=1}^{N} w_n \bar{J}(\pi_{n-1}) + w_n l_n(\pi_n)
\end{aligned}$$

therefore does not require extra Bregman divergence. Here PICCOLO with FTRL provides a natural generalization to online convex problems.

[13] Any stationary point will suffice.

$$
\begin{aligned}
&= w_1 \bar{J}(\pi_0) + \sum_{n=1}^{N-1} w_{n+1} \bar{J}(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
&= w_1 \bar{J}(\pi_0) + \sum_{n=1}^{N-1} w_{n+1} \bar{J}(\pi_{n-1}) + \sum_{n=1}^{N-1} w_{n+1} l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
&= (w_1 + w_2) \bar{J}(\pi_0) + \sum_{n=1}^{N-2} w_{n+2} \bar{J}(\pi_n) + \sum_{n=1}^{N-1} w_{n+1} l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
&= w_{1:N} \bar{J}(\pi_0) + \left(w_N l_1(\pi_1) + \sum_{n=1}^{2} w_{n+N-2} l_n(\pi_n) + \cdots + \sum_{n=1}^{N-1} w_{n+1} l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right) \\
&= w_{1:N} \bar{J}(\pi_0) + \left(w_N l_1(\pi_1) + \sum_{n=1}^{2} \frac{w_{n+N-2}}{w_n} w_n l_n(\pi_n) + \cdots + \sum_{n=1}^{N-1} \frac{w_{n+1}}{w_n} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right) \\
&\leq w_{1:N} \bar{J}(\pi_0) + \left(w_N l_1(\pi_1) + \frac{w_{N-1}}{w_1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + \frac{w_2}{w_1} \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right) \\
&= w_{1:N} \bar{J}(\pi_0) + \left(w_N l_1(\pi_1) + w_{N-1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + w_2 \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \right)
\end{aligned}
$$

where the inequality is due to the assumption on the weighting sequence.

We can further rearrange the second term in the final expression as

$$
\begin{aligned}
& w_N l_1(\pi_1) + w_{N-1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + w_2 \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
=& w_N \left(l_1(\pi_1) - \min_{\pi \in \Pi} l_1(\pi) + \min_{\pi \in \Pi} l_1(\pi) \right) \\
& + w_{N-1} \left(\sum_{n=1}^{2} w_n l_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{2} w_n l_n(\pi) + \min_{\pi \in \Pi} \sum_{n=1}^{2} w_n l_n(\pi) \right) \\
& + \cdots + \sum_{n=1}^{N} w_n l_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{N} w_n l_n(\pi) + \min_{\pi \in \Pi} \sum_{n=1}^{N} w_n l_n(\pi) \\
=& \sum_{n=1}^{N} w_{N-n+1} \left(\mathrm{regret}_n(f) + w_{1:n} \epsilon_n(f) \right)
\end{aligned}
$$

where the last equality is due to the definition of *static* regret and ϵ_n.

Likewise, we can also write the above expression in terms of *dynamic regret*

$$
\begin{aligned}
& w_N l_1(\pi_1) + w_{N-1} \sum_{n=1}^{2} w_n l_n(\pi_n) + \cdots + w_2 \sum_{n=1}^{N-1} w_n l_n(\pi_n) + \sum_{n=1}^{N} w_n l_n(\pi_n) \\
=& w_N \left(l_1(\pi_1) - \min_{\pi \in \Pi} l_1(\pi) + \min_{\pi \in \Pi} l_1(\pi) \right) \\
& + w_{N-1} \left(\sum_{n=1}^{2} w_n l_n(\pi_n) - \sum_{n=1}^{2} w_n \min_{\pi \in \Pi} l_n(\pi) + \sum_{n=1}^{2} \min_{\pi \in \Pi} w_n l_n(\pi) \right) \\
& + \cdots + \sum_{n=1}^{N} w_n l_n(\pi_n) - \sum_{n=1}^{N} \min_{\pi \in \Pi} w_n l_n(\pi) + \sum_{n=1}^{N} \min_{\pi \in \Pi} w_n l_n(\pi) \\
=& \sum_{n=1}^{N} w_{N-n+1} \left(\mathrm{regret}_n^d(l) + w_{1:n} \epsilon_n^d(l) \right)
\end{aligned}
$$

in which we define the weighted dynamic regret as

$$
\mathrm{regret}_n^d(l) = \sum_{m=1}^{n} w_m l_m(\pi_m) - \sum_{m=1}^{n} w_m \min_{\pi \in \Pi} l_m(\pi)
$$

and an expressive measure based on dynamic regret

$$
\epsilon_n^d = \frac{1}{w_{1:n}} \sum_{m=1}^{n} w_m \min_{\pi \in \Pi} l_m(\pi) \leq 0
$$

For stochastic problems, because π_n does not depends on $\tilde{l}_n$, the above bound applies to the performance in expectation. Specifically, let h_{n-1} denote all the random variables observed before making decision π_n and seeing $\tilde{l}_n$. As π_n is made independent of $\tilde{l}_n$, we have, for example,

$$
\begin{aligned}
\mathbb{E}[l_n(\pi_n)|h_{n-1}] &= \mathbb{E}[l_n(\pi_n)|h_{n-1}] - \mathbb{E}[l_n(\pi_n^*)|h_{n-1}] + \mathbb{E}[l_n(\pi_n^*)|h_{n-1}] \\
&= \mathbb{E}[\tilde{l}_n(\pi_n)|h_{n-1}] - \mathbb{E}[\tilde{l}_n(\pi_n^*)|h_{n-1}] + \mathbb{E}[l_n(\pi_n^*)|h_{n-1}] \\
&\leq \mathbb{E}[\tilde{l}_n(\pi_n) - \min_{\pi \in \Pi} \tilde{l}_n(\pi)|h_{n-1}] + \mathbb{E}[l_n(\pi_n^*)|h_{n-1}]
\end{aligned}
$$

where $\pi_n^* = \arg\min_{\pi \in \Pi} l_n(\pi)$. By applying a similar derivation as above recursively, we

can extend the previous deterministic bounds to bounds in expectation (for both the static or the dynamic regret case), proving the desired statement.

10.C The Basic Operations of Base Algorithms

We provide details of the abstract basic operations shared by different base algorithms. In general, the update rule of any base mirror-descent or FTRL algorithm can be represented in terms of the three basic operations

$$h \leftarrow \texttt{update}(h, H, g, w), \qquad H \leftarrow \texttt{adapt}(h, H, g, w), \qquad \pi \leftarrow \texttt{project}(h, H) \tag{10.17}$$

where `update` and `project` can be identified standardly, for mirror descent as,

$$\texttt{update}(h, H, g, w) = \arg\min_{\pi' \in \Pi} \langle wg, \pi' \rangle + B_H(\pi||h), \qquad \texttt{project}(h, H) = h \tag{10.18}$$

and for FTRL as,

$$\texttt{update}(h, H, g, w) = h + wg, \qquad \texttt{project}(h, H) = \arg\min_{\pi' \in \Pi} \langle h, \pi' \rangle + H(\pi') \tag{10.19}$$

We note that in the main text of this paper the operation `project` is omitted for simplicity, as it is equal to the identify map for mirror descent. In general, it represents the decoding from the abstract representation of the decision h to π. The main difference between and h and π is that h represents the sufficient information that defines the state of the base algorithm.

While `update` and `project` are defined standardly, the exact definition of `adapt` depends on the specific base algorithm. Particularly, `adapt` may depend also on whether the problem is weighted, as different base algorithms may handle weighted problems dif-

ferently. Based on the way weighted problems are handled, we roughly categorize the algorithms (in both mirror descent and FTRL families) into two classes: the *stationary* regularization class and the *non-stationary* regularization class. Here we provide more details into the algorithm-dependent `adapt` operation, through some commonly used base algorithms as examples.

Please see also Section 10.A for connection between PICCOLO and existing two-step algorithms, like optimistic mirror descent (Rakhlin and Sridharan, 2013).

10.C.1 Stationary Regularization Class

The `adapt` operation of these base algorithms features two major functions: 1) a moving-average adaptation and 2) a step-size adaption. The moving-average adaptation is designed to estimate some statistics G such that $\|g\|_* = O(G)$ (which is an important factor in regret bounds), whereas the step-size adaptation updates a scalar multiplier η according to the weight w to ensure convergence.

This family of algorithms includes basic mirror descent (Beck and Teboulle, 2003) and FTRL (McMahan, 2017; McMahan and Streeter, 2010) with a scheduled step size, and adaptive algorithms based on moving average e.g. RMSPROP (Tieleman and Hinton, 2012) ADADELTA (Zeiler, 2012), ADAM (Kingma and Ba, 2014), AMSGRAD (Reddi, Kale, and Kumar, 2018), and the adaptive NATGRAD we used in the experiments. Below we showcase how `adapt` is defined using some examples.

Basic mirror descent (Beck and Teboulle, 2003)

We define G to be some constant such that $G \geq \sup \|g_n\|_*$ and define

$$\eta_n = \frac{\eta}{1 + cw_{1:n}/\sqrt{n}}, \tag{10.20}$$

as a function of the iteration counter n, where $\eta > 0$ is a step size multiplier and $c > 0$ determines the decaying rate of the step size. The choice of hyperparameters η, c pertains to how far the optimal solution is from the initial condition, which is related to the size of Π. In implementation, `adapt` updates the iteration counter n and updates the multiplier η_n using w_n in (10.20).

Together (n, G, η_n) defines $H_n = R_n$ in the mirror descent update rule (10.7) through setting $R_n = \frac{G}{\eta_n} R$, where R is a strongly convex function. That is, we can write (10.7) equivalently as

$$\begin{aligned}
\pi_{n+1} &= \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + \frac{G}{\eta_n} B_R(\pi || \pi_n) \\
&= \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + B_{H_n}(\pi || \pi_n) \\
&= \texttt{update}(h_n, H_n, g_n, w_n)
\end{aligned}$$

When the weight is constant (i.e. $w_n = 1$), we can easily see this update rule is equivalent to the classical mirror descent with a step size $\frac{\eta/G}{1+c\sqrt{n}}$, which is the optimal step size (McMahan, 2017). For general $w_n = \Theta(n^p)$ with some $p > -1$, it can viewed as having an effective step size $\frac{w_n \eta_n}{G} = O(\frac{1}{G\sqrt{n}})$, which is optimal in the weighted setting. The inclusion of the constant G makes the algorithm invariant to the scaling of loss functions. But as the same G is used across all the iterations, the basic mirror descent is conservative.

Basic FTRL (McMahan, 2017)

We provide details of general FTRL

$$\pi_{n+1} = \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} \langle g_m, \pi \rangle + B_{r_m}(\pi || \pi_m) \tag{10.21}$$

where $B_{r_m}(\cdot || \pi_m)$ is a Bregman divergence centered at π_m.

We define, in the nth iteration, h_n, H_n, and $\texttt{project}$ of FTRL in (10.19) as

$$h_n = \sum_{m=1}^{n} w_m g_m, \qquad H_n(\pi) = \sum_{m=1}^{n} B_{r_m}(\pi||\pi_n), \qquad \texttt{project}(h, H) = \arg\min_{\pi' \in \Pi} \langle h, \pi' \rangle + H(\pi')$$

Therefore, we can see that $\pi_{n+1} = \texttt{project}(h_n, H_n)$ indeed gives the update (10.21):

$$\begin{aligned}
\pi_{n+1} &= \texttt{project}(h_n, H_n) \\
&= \texttt{project}(\sum_{m=1}^{n} w_m g_m, \sum_{m=1}^{n} B_{r_m}(\pi||\pi_n)) \\
&= \arg\min_{\pi \in \Pi} \sum_{m=1}^{n} \langle w_m g_m, \pi \rangle + B_{r_m}(\pi||\pi_m)
\end{aligned}$$

For the basic FTRL, the $\texttt{adapt}$ operator is similar to the basic mirror descent, which uses a constant G and updates the memory (n, η_n) using (10.20). The main differences are how (G, η_n) is mapped to H_n and that the basic FTRL updates H_n also using h_n (i.e. π_n). Specifically, it performs $H_n \leftarrow \texttt{adapt}(h_n, H_{n-1}, g_n, w_n)$ through the following:

$$H_n(\cdot) = H_{n-1}(\cdot) + B_{r_n}(\cdot||\pi_n)$$

where following (McMahan, 2017) we set

$$B_{r_n}(\pi||\pi_n) = G(\frac{1}{\eta_n} - \frac{1}{\eta_{n-1}})B_R(\pi||\pi_n)$$

and η_n is updated using some scheduled rule.

One can also show that the choice of η_n scheduling in (10.20) leads to an optimal regret. When the problem is uniformly weighted (i.e. $w_n = 1$), this gives exactly the update rule in (McMahan, 2017). For general $w_n = \Theta(n^p)$ with $p > -1$, a proof of optimality can be found, for example, in the appendix of (Cheng et al., 2019b).

ADAM *(Kingma and Ba, 2014) and* AMSGRAD *(Reddi, Kale, and Kumar, 2018)*

As a representing mirror descent algorithm that uses moving-average estimates, ADAM keeps in the memory of the statistics of the first-order information that is provided in `update` and `adapt`. Here we first review the standard description of ADAM and then show how it is summarized in

$$H_n = \texttt{adapt}(h_n, H_{n-1}, g_n, w_n), \qquad h_{n+1} = \texttt{update}(h_n, H_n, g_n, w_n) \tag{10.8}$$

using properly constructed `update`, `adapt`, and `project` operations.

The update rule of ADAM proposed by Kingma and Ba (2014) is originally written as, for $n \geq 1$,[14]

$$\begin{aligned} m_n &= \beta_1 m_{n-1} + (1-\beta_1) g_n \\ v_n &= \beta_2 v_{n-1} + (1-\beta_2) g_n \odot g_n \\ \hat{m}_n &= m_n / (1-\beta_1^n) \\ \hat{v}_n &= v_n / (1-\beta_2^n) \\ \pi_{n+1} &= \pi_n - \eta_n \hat{m}_n \oslash (\sqrt{\hat{v}_n} + \epsilon) \end{aligned} \tag{10.22}$$

where $\eta_n > 0$ is the step size, $\beta_1, \beta_2 \in [0, 1)$ (default $\beta_1 = 0.9$ and $\beta_2 = 0.999$) are the mixing rate, and $0 < \epsilon \ll 1$ is some constant for stability (default $\epsilon = 10^{-8}$), and $m_0 = v_0 = 0$. The symbols $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively. The third and the forth steps are designed to remove the 0-bias due to running moving averages starting from 0.

The above update rule can be written in terms of the three basic operations. First, we define the memories $h_n = (m_n, \pi_n)$ for policy and (v_n, η_n, n) for regularization that is

[14] We shift the iteration index so it conforms with our notation in online learning, in which π_1 is the initial policy before any update.

defined as

$$H_n(\pi) = \frac{1}{2\eta_n}\pi^\top(\mathrm{diag}(\sqrt{\hat{v}_n}) + \epsilon I)\pi \tag{10.23}$$

where $\hat{v}_n$ is defined in the original ADAM equation in (10.22).

The `adapt` operation updates the memory to (v_n, η_n, n) in the step

$$H_n \leftarrow \texttt{adapt}(h_n, H_{n-1}, g_n, w_n)$$

It updates the iteration counter n and η_n in the same way in the basic mirror descent using (10.20), and update v_n (which along with n defines $\hat{v}_n$ used in (10.23)) using the original ADAM equation in (10.22).

For `update`, we slightly modify the definition of `update` in (10.18) (replacing g_n with $\hat{m}_n$) to incorporate the moving average and write

$$\texttt{update}(h_n, H_n, g_n, w_n) = \arg\min_{\pi' \in \Pi} \langle w_n \hat{m}_n, \pi' \rangle + B_{H_n}(\pi' || \pi) \tag{10.24}$$

where m_n and $\hat{m}_n$ are defined the same as in the original ADAM equations in (10.22). One can verify that, with these definitions, the update rule in (10.8) is equivalent to the update rule (10.22), when the weight is uniform (i.e. $w_n = 1$).

Here the $\sqrt{\hat{v}_n}$ plays the role of G as in the basic mirror descent, which can be viewed as an estimate of the upper bound of $\|g_n\|_*$. ADAM achieves a better performance because a coordinate-wise online estimate is used. With this equivalence in mind, we can easily deduct that using the same scheduling of η_n as in the basic mirror descent would achieve an optimal regret (cf. (Kingma and Ba, 2014; Reddi, Kale, and Kumar, 2018)). We note that ADAM may fail to converge in some particular problems due to the moving average (Reddi, Kale, and Kumar, 2018). AMSGRAD (Reddi, Kale, and Kumar, 2018) modifies the moving average and uses strictly increasing estimates. However in practice AMSGRAD behaves

more conservatively.

For weighted problems, we note one important nuance in our definition above: it separates the weight w_n from the moving average and considers w_n as part of the η_n update, because the growth of w_n in general can be much faster than the rate the moving average converges. In other words, the moving average can only be used to estimate a stationary property, not a time-varying one like w_n. Hence, we call this class of algorithms, the *stationary* regularization class.

Adaptive NATGRAD

Given first-order information g_n and weight w_n, we consider an update rule based on Fisher information matrix:

$$\pi_{n+1} = \arg\min_{\pi \in \Pi} \langle w_n g_n, \pi \rangle + \frac{\sqrt{\hat{G}_n}}{2\eta_n} (\pi - \pi_n)^\top F_n (\pi - \pi_n) \tag{10.25}$$

where F_n is the Fisher information matrix of policy π_n (Amari, 1998) and $\hat{G}_n$ is an adaptive multiplier for the step size which we will describe. When $\hat{G}_n = 1$, the update in (10.25) gives the standard natural gradient descent update with step size η_n (Kakade, 2002) .

The role of $\hat{G}_n$ is to adaptively and *slowly* changes the step size to minimize $\sum_{n=1}^{N} \frac{\eta_n}{\sqrt{\hat{G}_n}} \|g_n\|^2_{F_n,*}$, which plays an important part in the regret bound (see Section 10.5, Section 10.F, and e.g. (McMahan, 2017) for details). Following the idea in ADAM, we update $\hat{G}_n$ by setting (with $G_0 = 0$)

$$\begin{aligned} G_n &= \beta_2 G_n + (1 - \beta_2) \frac{1}{2} g_n^\top F_n^{-1} g_n \\ \hat{G}_n &= G_n / (1 - \beta_2^n) \end{aligned} \tag{10.26}$$

similar to the concept of updating v_n and $\hat{v}_n$ in ADAM in (10.22), and update η_n in the same way as in the basic mirror descent using (10.20). Consequently, this would also lead to a regret like ADAM but in terms of a different local norm.

The `update` operation of adaptive NATGRAD is defined standardly in (10.7) (as used in the experiments). The `adapt` operation updates n and η_n like in ADAM and updates G_n through (10.26).

10.C.2 Non-Stationary Regularization Class

The algorithms in the non-stationary regularization class maintains a regularization that is increasing over the number of iterations. Notable examples of this class include ADAGRAD (Duchi, Hazan, and Singer, 2011) and ONLINE NEWTON STEP (Hazan, Agarwal, and Kale, 2007), and its regularization function is updated by applying BTL in a secondary online learning problem whose loss is an upper bound of the original regret (see (Gupta, Koren, and Singer, 2017) for details). Therefore, compared with the previous stationary regularization class, the adaption property of η_n and G_n exchanges: η_n here becomes constant and G_n becomes time-varying. This will be shown more clearly in the ADAGRAD example below. We note while these algorithms are designed to be optimal in the convex, they are often too conservative (e.g. decaying the step size too fast) for non-convex problems.

ADAGRAD

The update rule of the diagonal version of ADAGRAD in (Duchi, Hazan, and Singer, 2011) is given as

$$\begin{aligned} G_n &= G_{n-1} + \operatorname{diag}(g_n \odot g_n) \\ \pi_{n+1} &= \arg\min_{\pi \in \Pi} \langle g_n, \pi \rangle + \frac{1}{2\eta}(\pi - \pi_n)^\top (\epsilon I + G_n)^{1/2}(\pi - \pi_n) \end{aligned} \tag{10.27}$$

where $G_0 = 0$ and $\eta > 0$ is a constant. ADAGRAD is designed to be optimal for online linear optimization problems. Above we provide the update equations of its mirror descent formulation in (10.27); a similar FTRL is also available (again the difference only happens when Π is constrained).

In terms of our notation, its `update` and `project` are defined standardly as in (10.18), i.e.

$$\texttt{update}(h_n, H_n, g_n, w_n) = \arg\min_{\pi' \in \Pi} \langle w_n g_n, \pi' \rangle + B_{H_n}(\pi' || \pi_n) \tag{10.28}$$

and its `adapt` essentially only updates G_n:

$$\texttt{adapt}(h_n, H_{n-1}, g_n, w_n) : G_n = G_{n-1} + \mathrm{diag}(w_n g_n \odot w_n g_n)$$

where the regularization is defined the updated G_n and the constant η as

$$H_n(\pi) = \frac{1}{2\eta} \pi^\top (\epsilon I + G_n)^{1/2} \pi.$$

One can simply verify the above definitions of `update` and `adapt` agrees with (10.27).

10.D A Practical Variation of PICCOLO

In Section 10.4.2, we show that, given a base algorithm in mirror descent/FTRL, PICCOLO generates a new first-order update rule by recomposing the three basic operations into

$$h_n = \texttt{update}(\hat{h}_n, H_{n-1}, \hat{g}_n, w_n) \qquad \text{[Prediction]} \tag{10.29}$$

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, H_{n-1}, e_n, w_n) \\ \hat{h}_{n+1} &= \texttt{update}(h_n, H_n, e_n, w_n) \end{aligned} \qquad \text{[Correction]} \tag{10.30}$$

where $e_n = g_n - \hat{g}_n$ and $\hat{g}_n$ is an estimate of g_n given by a predictive model Φ_n.

Here we propose a slight variation which introduces another operation `shift` inside the

Prediction Step. This leads to the new set of update rules:

$$\begin{aligned} \hat{H}_n &= \texttt{shift}(\hat{h}_n, H_{n-1}) \\ h_n &= \texttt{update}(\hat{h}_n, \hat{H}_n, \hat{g}_n, w_n) \end{aligned} \qquad \text{[Prediction]} \tag{10.31}$$

$$\begin{aligned} H_n &= \texttt{adapt}(h_n, \hat{H}_n, e_n, w_n) \\ \hat{h}_{n+1} &= \texttt{update}(h_n, H_n, e_n, w_n) \end{aligned} \qquad \text{[Correction]} \tag{10.32}$$

The new `shift` operator additionally changes the regularization based on $\hat{h}_n$ the current representation of the policy in the Prediction Step, *independent* of the predicted gradient $\hat{g}_n$ and weight w_n. The main purpose of including this additional step is to deal with numerical difficulties, such as singularity of H_n. For example, in natural gradient descent, the Fisher information of some policy can be close to being singular along the direction of the gradients that are evaluated at different policies. As a result, in the original Prediction Step of PICCOLO, H_{n-1} which is evaluated at π_{n-1} might be singular in the direction of $\hat{g}_n$ which is evaluated $\hat{\pi}_n$.

The new operator `shift` brings in an extra degree of freedom to account for such issue. Although from a theoretical point of view (cf. Section 10.F) the use of `shift` would only increase regrets and should be avoided if possible, in practice, its merits in handling numerical difficulties can out weight the drawback. Because `shift` does not depend on the size of $\hat{g}_n$ and w_n, the extra regrets would only be proportional to $O(\sum_{n=1}^{N} \|\pi_n - \hat{\pi}_n\|_n)$, which can be smaller than other terms in the regret bound (see Section 10.F).

10.E Example: PICCOLOing Natural Gradient Descent

We give an alternative example to illustrate how one can use the above procedure to "PICCOLO" a base algorithm into a new algorithm. Here we consider the adaptive natural gradient descent rule in Section 10.C as the base algorithm, which (given first-order

information g_n and weight w_n) updates the policy through

$$\pi_{n+1} = \arg\min_{\pi\in\Pi} \langle w_n g_n, \pi\rangle + \frac{\sqrt{\hat{G}_n}}{2\eta_n}(\pi - \pi_n)^\top F_n(\pi - \pi_n) \tag{10.33}$$

where F_n is the Fisher information matrix of policy π_n (Amari, 1998), η_n a scheduled learning rate, and $\hat{G}_n$ is an adaptive multiplier for the step size which we will shortly describe. When $\hat{G}_n = 1$, the update in (10.33) gives the standard natural gradient descent update with step size η_n (Kakade, 2002) .

The role of $\hat{G}_n$ is to adaptively and *slowly* changes the step size to minimize $\sum_{n=1}^N \frac{\eta_n}{\sqrt{G_n}}\|g_n\|^2_{F_n,*}$, which plays an important part in the regret bound (see Section 10.5, Section 10.F, and e.g. (McMahan, 2017) for details). To this end, we update $\hat{G}_n$ by setting (with $G_0 = 0$)

$$G_n = \beta_2 G_{n-1} + (1-\beta_2)\tfrac{1}{2} g_n^\top F_n^{-1} g_n, \qquad \hat{G}_n = G_n/(1-\beta_2^n) \tag{10.34}$$

similar to the moving average update rule in ADAM, and update η_n in the same way as in the basic mirror descent algorithm (e.g. $\eta_n = O(1/\sqrt{n})$). As a result, this leads to a similar regret like ADAM with $\beta_1 = 0$, but in terms of a local norm specified by the Fisher information matrix.

Now, let's see how to PICCOLO the adaptive natural gradient descent rule above. First, it is easy to see that the adaptive natural gradient descent rule is an instance of mirror descent (with $h_n = \pi_n$ and $H_n(g) = \frac{\sqrt{\hat{G}_n}}{2\eta_n} g^\top F_n g$), so the `update` and `project` operations are defined in the standard way, as in Section 10.4.2. The `adapt` operation updates the iteration counter n, the learning rate η_n, and updates $\hat{G}_n$ through (10.34).

To be more specific, let us explicitly write out the Prediction Step and the Correction Step of the PICCOLOed adaptive natural gradient descent rule in closed form as below: e.g. if $\eta_n = \frac{1}{\sqrt{n}}$, then we can write them as

$$\text{[Prediction]} \qquad \pi_n = \arg\min_{\pi\in\Pi} \langle w_n \hat{g}_n, \pi\rangle + \frac{\sqrt{\hat{G}_{n-1}}}{2\eta_{n-1}}(\pi - \hat{\pi}_n)^\top F_{n-1}(\pi - \hat{\pi}_n)$$

[Correction]

$$\eta_n = 1/\sqrt{n}$$
$$G_n = \beta_2 G_{n-1} + (1-\beta_2)\tfrac{1}{2} g_n^\top F_n^{-1} g_n$$
$$\hat{G}_n = G_n/(1-\beta_2^n)$$
$$\hat{\pi}_{n+1} = \arg\min_{\pi\in\Pi} \langle w_n e_n, \pi\rangle + \frac{\sqrt{\hat{G}_n}}{2\eta_n}(\pi-\pi_n)^\top F_n(\pi-\pi_n)$$

10.F Regret Analysis of PICCOLO

The main idea of PICCOLO is to achieve optimal performance in predictable online learning problems by *reusing* existing adaptive, optimal first-order algorithms that are designed for adversarial online learning problems. This is realized by the reduction techniques presented in this section.

Here we prove the performance of PICCOLO in general predictable online learning problems, independent of the context of policy optimization. In Section 10.F.1, we first show an elegant reduction from predictable problems to adversarial problems. Then we prove Theorem 10.5.1 in Section 10.F.2, showing how the optimal regret bound for predictable linear problems can be achieved by PICCOLOing mirror descent and FTRL algorithms. Note that we will abuse the notation l_n to denote the per-round losses in this general setting.

10.F.1 Reduction from Predictable Online Learning to Adversarial Online Learning

Consider a predictable online learning problem with per-round losses $\{l_n\}$. Suppose in round n, before playing π_n and revealing l_n, we have access to some prediction of l_n, called $\hat{l}_n$. In particular, we consider the case where $\hat{l}_n(\pi) = \langle \hat{g}_n, \pi\rangle$ for some vector $\hat{g}_n$. Running an (adaptive) online learning algorithm designed for the general adversarial setting is not optimal here, as its regret would be in $O(\sum_{n=1}^N \|\nabla l_n\|_{n,*}^2)$, where $\|\cdot\|_n$ is some local norm chosen by the algorithm and $\|\cdot\|_{n,*}$ is its dual norm. Ideally, we would only want to pay

for the information that is unpredictable. Specifically, we wish to achieve an optimal regret in $O(\sum_{n=1}^{N} \|\nabla l_n - \nabla \hat{l}_n\|_{n,*}^2)$ instead (Rakhlin and Sridharan, 2012).

To achieve the optimal regret bound yet without referring to specialized, nested two-step algorithms (e.g. mirror-prox Juditsky, Nemirovski, and Tauvel, 2011, optimistic mirror descent (Rakhlin and Sridharan, 2013), FTRL-prediction Rakhlin and Sridharan, 2012), we consider decomposing a *predictable* problem with N rounds into an *adversarial* problem with $2N$ rounds:

$$\sum_{n=1}^{N} l_n(\pi_n) = \sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) \tag{10.35}$$

where $\delta_n = l_n - \hat{l}_n$. Therefore, we can treat the predictable problem as a new adversarial online learning problem with a loss sequence $\hat{l}_1, \delta_1, \hat{l}_2, \delta_2, \ldots, \hat{l}_N, \delta_N$ and consider solving this new problem with some standard online learning algorithm designed for the adversarial setting.

Before analysis, we first introduce a new decision variable $\hat{\pi}_n$ and denote the decision sequence in this new problem as $\hat{\pi}_1, \pi_1, \hat{\pi}_2, \pi_2, \ldots, \hat{\pi}_N, \pi_N$, so the definition of the variables are consistent with that in the problem before. Because this new problem is unpredictable, the optimal regret of this new decision sequence is

$$\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{N} \hat{l}_n(\pi) + \delta_n(\pi) = O(\sum_{n=1}^{N} \|\nabla \hat{l}_n\|_{n,*}^2 + \|\nabla \delta_n\|_{n+1/2,*}^2) \tag{10.36}$$

where the subscript $n + 1/2$ denotes the extra round due to the reduction.

At first glance, our reduction does not meet the expectation of achieving regret in $O(\sum_{n=1}^{N} \|\nabla l_n - \nabla \hat{l}_n\|_{n,*}^2) = O(\sum_{n=1}^{N} \|\nabla \delta_n\|_{n,*}^2)$. However, we note that the regret for

the new problem is too loose for the regret of the original problem, which is

$$\sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{N} \hat{l}_n(\pi) + \delta_n(\pi)$$

where the main difference is that originally we care about $\hat{l}_n(\pi_n)$ rather than $\hat{l}_n(\hat{\pi}_n)$. Specifically, we can write

$$\begin{aligned}\sum_{n=1}^{N} l_n(\pi_n) &= \sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) \\ &= \left(\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n)\right) + \left(\sum_{n=1}^{N} \hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)\right)\end{aligned}$$

Therefore, if the update rule for generating the decision sequence $\hat{\pi}_1, \pi_1, \hat{\pi}_2, \pi_2, \ldots, \hat{\pi}_N, \pi_N$ contributes sufficient negativity in the term $\hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)$ compared with the regret of the new adversarial problem, then the regret of the original problem can be smaller than (10.36). This is potentially possible, as π_n is made after $\hat{l}_n$ is revealed. Especially, in the fixed-point formulation of PICCOLO, π_n and $\hat{l}_n$ can be decided simultaneously.

In the next section, we show that when the base algorithm, which is adopted to solve the new adversarial problem given by the reduction, is in the family of mirror descent and FTRL. Then the regret bound of PICCOLO with respect to the original predictable problem is optimal.

10.F.2 Optimal Regret Bounds for Predictable Problems

We show that if the base algorithm of PICCOLO belongs to the family of optimal mirror descent and FTRL designed for adversarial problems, then PICCOLO can achieve the optimal regret of predictable problems. In this subsection, we assume the loss sequence is linear, i.e. $l_n(\pi) = \langle g_n, \pi \rangle$ for some g_n, and the results are summarized as Theorem 10.5.1 in the main paper (in a slightly different notation).

Mirror Descent

First, we consider mirror descent as the base algorithm. In this case, we can write the PICCOLO update rule as

$$\pi_n = \arg\min_{\pi \in \Pi} \left\langle \nabla \hat{l}_n(\hat{\pi}_n), x \right\rangle + B_{H_{n-1}}(\pi || \hat{\pi}_n) \qquad \text{[Prediction]}$$

$$\hat{\pi}_{n+1} = \arg\min_{\pi \in \Pi} \langle \nabla \delta_n(\pi_n), \pi \rangle + B_{H_n}(\pi || \pi_n) \qquad \text{[Correction]}$$

where H_n can be updated based on $e_n \coloneqq \nabla \delta_n(\pi_n) = \nabla l_n(\pi_n) - \nabla \hat{l}_n(\hat{\pi}_n)$ (recall by definition $\nabla l_n(\pi_n) = g_n$ and $\nabla \hat{l}_n(\hat{\pi}_n) = \nabla \hat{l}_n(\pi_n) = \hat{g}_n$). Notice that in the Prediction Step, PICCOLO uses the regularization from the previous Correction Step.

To analyze the performance, we use a lemma of the mirror descent's properties. The proof is a straightforward application of the optimality condition of the proximal map Nesterov, 2013. We provide a proof here for completeness.

Lemma 10.F.1. *Let $\mathcal{K}$ be a convex set. Suppose R is 1-strongly convex with respect to norm $\|\cdot\|$. Let g be a vector in some Euclidean space and let*

$$y = \arg\min_{z \in \mathcal{K}} \langle g, z \rangle + \frac{1}{\eta} B_R(z||x)$$

Then for all $z \in \mathcal{K}$

$$\eta \langle g, y - z \rangle \leq B_R(z||x) - B_R(z||y) - B_R(y||x) \tag{10.37}$$

which implies

$$\eta \langle g, x - z \rangle \leq B_R(z||x) - B_R(z||y) + \frac{\eta^2}{2} \|g\|_*^2 \tag{10.38}$$

Proof. Recall the definition $B_R(z||x) = R(z) - R(x) - \langle \nabla R(x), z - x \rangle$. The optimality

of the proximal map can be written as

$$\langle \eta g + \nabla R(y) - \nabla R(x), y - z \rangle \leq 0, \qquad \forall z \in \mathcal{K}$$

By rearranging the terms, we can rewrite the above inequality in terms Bregman divergences as follows and derive the first inequality (10.37):

$$\begin{aligned}
\langle \eta g, y - z \rangle &\leq \langle \nabla R(x) - \nabla R(y), y - z \rangle \\
&= B_R(z||x) - B_R(z||y) + \langle \nabla R(x) - \nabla R(y), y \rangle - \langle \nabla R(x), x \rangle \\
&\quad + \langle \nabla R(y), y \rangle + R(x) - R(y) \\
&= B_R(z||x) - B_R(z||y) + \langle \nabla R(x), y - x \rangle + R(x) - R(y) \\
&= B_R(z||x) - B_R(z||y) - B_R(y||x)
\end{aligned}$$

The second inequality is the consequence of (10.37). First, we rewrite (10.37) as

$$\langle \eta g, x - z \rangle = B_R(z||x) - B_R(z||y) - B_R(y||x) + \langle \eta g, x - y \rangle$$

Then we use the fact that B_R is 1-strongly convex with respect to $\|\cdot\|$, which implies

$$-B_R(y||x) + \langle \eta g, x - y \rangle \leq -\frac{1}{2}\|x - y\|^2 + \langle \eta g, x - y \rangle \leq \frac{\eta^2}{2}\|g\|_*^2$$

Combining the two inequalities yields (10.38). ■

Lemma 10.F.1 is usually stated with (10.38), which concerns the decision made before seeing the per-round loss (as in the standard adversarial online learning setting). Here, we additionally concern $\hat{l}_n(\pi_n)$, which is the decision made after seeing $\hat{l}_n$, so we need a tighter bound (10.37).

Now we show that the regret bound of PICCOLO in the predictable linear problems when the base algorithm is mirror descent.

Proposition 10.F.1. *Assume the base algorithm of* PICCOLO *is mirror descent satisfying the Assumption 10.5.1. Let* $g_n = \nabla l_n(\pi_n)$ *and* $e_n = g_n - \hat{g}_n$*. Then it holds that, for any* $\pi \in \Pi$*,*

$$\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle \leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2} \|e_n\|_{*,n}^2 - \frac{1}{2} \|\pi_n - \hat{\pi}_n\|_{n-1}^2$$

Proof. Suppose R_n, which is defined by H_n, is 1-strongly convex with respect to $\|\cdot\|_n$. Then by Lemma 10.F.1, we can write, for all $\pi \in \Pi$,

$$\begin{aligned} w_n \langle g_n, \pi_n - \pi \rangle &= w_n \langle \hat{g}_n, \pi_n - \pi \rangle + w_n \langle e_n, \pi_n - \pi \rangle \\ &\leq B_{R_{n-1}}(\pi || \hat{\pi}_n) - B_{R_{n-1}}(\pi || \pi_n) - B_{R_{n-1}}(\pi_n || \hat{\pi}_n) \\ &\quad + B_{R_n}(\pi || \pi_n) - B_{R_n}(\pi || \hat{\pi}_{n+1}) + \frac{w_n^2}{2} \|e_n\|_{*,n}^2 \end{aligned} \tag{10.39}$$

where we use (10.37) for $\hat{g}_n$ and (10.38) for the loss e_n.

To show the regret bound of the original (predictable) problem, we first notice that

$$\begin{aligned} &\sum_{n=1}^{N} B_{R_{n-1}}(\pi || \hat{\pi}_n) - B_{R_{n-1}}(\pi || \pi_n) + B_{R_n}(\pi || \pi_n) - B_{R_n}(\pi || \hat{\pi}_{n+1}) \\ &= B_{R_0}(\pi || \hat{\pi}_1) - B_{R_N}(\pi || \hat{\pi}_{N+1}) + \sum_{n=1}^{N} B_{R_{n-1}}(\pi || \hat{\pi}_n) - B_{R_{n-1}}(\pi || \pi_n) + B_{R_n}(\pi || \pi_n) - B_{R_{n-1}}(\pi || \hat{\pi}_n) \\ &= B_{R_0}(\pi || \hat{\pi}_1) - B_{R_N}(\pi || \hat{\pi}_{N+1}) + \sum_{n=1}^{N} B_{R_n}(\pi || \pi_n) - B_{R_{n-1}}(\pi || \pi_n) \leq M_N \end{aligned}$$

where the last inequality follows from the assumption on the base algorithm. Therefore, by telescoping the inequality in (10.39) and using the strong convexity of R_n, we get

$$\begin{aligned} \sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle &\leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2} \|e_n\|_{*,n}^2 - B_{R_{n-1}}(\pi_n || \hat{\pi}_n) \\ &\leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2} \|e_n\|_{*,n}^2 - \frac{1}{2} \|\pi_n - \hat{\pi}_n\|_{n-1}^2 \end{aligned}$$

■

Follow-the-Regularized-Leader

We consider another type of base algorithm, FTRL, which is mainly different from mirror descent in the way that constrained decision sets are handled (McMahan, 2017). In this case, the exact update rule of PICCOLO can be written as

$$\pi_n = \arg\min_{\pi\in\Pi} \langle w_n \hat{g}_n, \pi\rangle + \sum_{m=1}^{n-1} \langle w_m g_m, \pi\rangle + B_{r_m}(\pi||\pi_m) \qquad \text{[Prediction]}$$

$$\hat{\pi}_{n+1} = \arg\min_{\pi\in\Pi} \sum_{m=1}^{n} \langle w_m g_m, \pi\rangle + B_{r_m}(\pi||\pi_m) \qquad \text{[Correction]}$$

From the above equations, we verify that MOBIL (Cheng et al., 2019b) is indeed a special case of PICCOLO, when the base algorithm is FTRL.

We show PICCOLO with FTRL has the following guarantee.

Proposition 10.F.2. *Assume the base algorithm of* PICCOLO *is FTRL satisfying the Assumption 10.5.1. Then it holds that, for any* $\pi \in \Pi$*,*

$$\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi\rangle \le M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$$

We show the above results of PICCOLO using a different technique from (Cheng et al., 2019b). Instead of developing a specialized proof like they do, we simply use the properties of FTRL on the $2N$-step new adversarial problem!

To do so, we recall some facts of the base algorithm FTRL. First, FTRL in (10.21) is equivalent to Follow-the-Leader (FTL) on a surrogate problem with the per-round loss is $\langle g_n, \pi\rangle + B_{r_n}(\pi||\pi_n)$. Therefore, the regret of FTRL can be bounded by the regret of FTL in the surrogate problem plus the size of the additional regularization $B_{r_n}(\pi||\pi_n)$. Second, we recall a standard techniques in proving FTL, called Strong FTL Lemma (see e.g. (McMahan, 2017)), which is proposed for *adversarial* online learning.

Lemma 10.F.2 (Strong FTL Lemma (McMahan, 2017))**.** *For any sequence* $\{\pi_n \in \Pi\}$ *and*

$\{l_n\}$,

$$\text{regret}_N(l) := \sum_{n=1}^{N} l_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^{N} l_n(\pi) \leq \sum_{n=1}^{N} l_{1:n}(\pi_n) - l_{1:n}(\pi_n^\star)$$

where $\pi_n^\star \in \arg\min_{\pi \in \Pi} l_{1:n}(\pi)$.

Using the decomposition idea above, we show the performance of PICCOLO following sketch below: first, we show a bound on the regret in the surrogate predictable problem with per-round loss $\langle g_n, \pi \rangle + B_{r_n}(\pi || \pi_n)$; second, we derive the bound for the original predictable problem with per-round loss $\langle g_n, \pi \rangle$ by considering the effects of $B_{r_n}(\pi || \pi_n)$. We will prove the first step by applying FTL on the transformed $2N$-step adversarial problem of the original N-step predictable surrogate problem and then showing that PICCOLO achieves the optimal regret in the original N-step predictable surrogate problem. Interestingly, we recover the bound in the stronger FTL Lemma (Lemma 10.F.3) by Cheng et al. (2019b), which they suggest is necessary for proving the improved regret bound of their FTRL-prediction algorithm (MOBIL).

Lemma 10.F.3 (Stronger FTL Lemma (Cheng et al., 2019b))**.** *For any sequence* $\{\pi_n\}$ *and* $\{l_n\}$,

$$\text{regret}_N(l) = \sum_{n=1}^{N} l_{1:n}(\pi_n) - l_{1:n}(\pi_n^\star) - \Delta_n$$

where $\Delta_{n+1} := l_{1:n}(\pi_{n+1}) - l_{1:n}(\pi_n^\star) \geq 0$ *and* $\pi_n^\star \in \arg\min_{\pi \in \Pi} l_{1:n}(\pi)$.

Our new reduction-based regret bound is presented below.

Proposition 10.F.3. *Let* $\{l_n\}$ *be a predictable loss sequence with predictable information* $\{\hat{l}_n\}$. *Suppose the decision sequence* $\hat{\pi}_1, \pi_1, \hat{\pi}_2, \ldots, \hat{\pi}_N, \pi_N$ *is generated by running FTL on the transformed adversarial loss sequence* $\hat{l}_1, \delta_1, \hat{l}_2, \ldots, \hat{l}_N, \delta_N$*, then the bound in the Stonger FTL Lemma holds. That is,* $\text{regret}_N(l) \leq \sum_{n=1}^{N} l_{1:n}(\pi_n) - l_{1:n}(\pi_n^\star) - \Delta_n$*, where* $\Delta_{n+1} := l_{1:n}(\pi_{n+1}) - l_{1:n}(\pi_n^\star) \geq 0$ *and* $\pi_n^\star \in \arg\min_{\pi \in \Pi} l_{1:n}(\pi)$.

Proof. First, we transform the loss sequence and write

$$\sum_{n=1}^{N} l_n(\pi_n) = \sum_{n=1}^{N} \hat{l}_n(\pi_n) + \delta_n(\pi_n) = \left(\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n)\right) + \left(\sum_{n=1}^{N} \hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)\right)$$

Then we apply standard Strong FTL Lemma on the new adversarial problem in the left term.

$$\begin{aligned}
&\sum_{n=1}^{N} \hat{l}_n(\hat{\pi}_n) + \delta_n(\pi_n) \\
&\leq \sum_{n=1}^{N} (\hat{l} + \delta)_{1:n}(\pi_n) - \min_{\pi \in \Pi} (\hat{l} + \delta)_{1:n}(\pi) + \sum_{n=1}^{N} ((\hat{l} + \delta)_{1:n-1} + \hat{l}_n)(\hat{\pi}_n) - \min_{\pi \in \Pi} ((\hat{l} + \delta)_{1:n-1} + \hat{l}_n)(\pi) \\
&= \sum_{n=1}^{N} l_{1:n}(\pi_n) - \min_{\pi \in \Pi} l_{1:n}(\pi) + \sum_{n=1}^{N} (l_{1:n-1} + \hat{l}_n)(\hat{\pi}_n) - (l_{1:n-1} + \hat{l}_n)(\pi_n)
\end{aligned}$$

where the first inequality is due to Strong FTL Lemma and the second equality is because FTL update assumption.

Now we observe that if we add the second term above and $\sum_{n=1}^{N} \hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)$ together, we have

$$\begin{aligned}
&\sum_{n=1}^{N} (l_{1:n-1} + \hat{l}_n)(\hat{\pi}_n) - (l_{1:n-1} + \hat{l}_n)(\pi_n) + (\hat{l}_n(\pi_n) - \hat{l}_n(\hat{\pi}_n)) \\
&= \sum_{n=1}^{N} (l_{1:n-1})(\hat{\pi}_n) - l_{1:n-1}(\pi_n) = \Delta_n
\end{aligned}$$

Thus, combing previous two inequalities, we have the bound in the Stronger FTL Lemma:

$$\sum_{n=1}^{N} l_n(\pi_n) \leq \sum_{n=1}^{N} l_{1:n}(\pi_n) - \min_{\pi \in \Pi} l_{1:n}(\pi) - \Delta_n \qquad \blacksquare$$

Using Proposition 10.F.3, we can now bound the regret of PICCOLO in Proposition 10.F.2 easily.

Proof of Proposition 10.F.2. Suppose $\sum_{m=1}^{n} B_{r_m}(\cdot||\pi_m)$ is 1-strongly convex with respect

to some norm $\|\cdot\|_n$. Let $f_n = \langle w_n g_n, \pi_n \rangle + B_{r_n}(\pi||\pi_m)$. Then by a simple convexity analysis (see e.g. see (McMahan, 2017)) and Proposition 10.F.3, we can derive

$$\begin{aligned}\text{regret}_N(f) &\leq \sum_{n=1}^{N} (f_{1:n}(\pi_n) - \min_{\pi\in\Pi} f_{1:n}(\pi)) - (f_{1:n-1}(\pi_n) - f_{1:n-1}(\hat{\pi}_n)) \\ &\leq \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{n,*}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2\end{aligned}$$

Finally, because r_n is proximal (i.e. $B_{r_n}(\pi_n||\pi_n) = 0$), we can bound the original regret: for any $\pi \in \Pi$, it satisfies that

$$\begin{aligned}\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle &\leq \sum_{n=1}^{N} f_n(\pi_n) - f_n(\pi) + B_{r_n}(\pi||\pi_n) \\ &\leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2\end{aligned}$$

where we use Assumption 10.5.1 and the bound of $\text{regret}_N(f)$ in the second inequality. ■

10.G Policy Optimization Analysis of PICCOLO

In this section, we discuss how to interpret the bound given in Theorem 10.5.1

$$\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle \leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$$

in the context of policy optimization and derive the bound

$$\mathbb{E}\left[\sum_{n=1}^{N} \langle w_n g_n, \pi_n - \pi \rangle\right] \leq O(1) + C_{\Pi,\Phi}\frac{w_{1:N}}{\sqrt{N}} \tag{10.40}$$

We will also discuss how model learning can further help minimize the regret bound later in Section 10.G.4.

10.G.1 Assumptions

We introduce some assumptions to characterize the sampled gradient g_n. Recall $g_n = \nabla \tilde{l}_n(\pi_n)$.

Assumption 10.G.1. $\|\mathbb{E}[g_n]\|_*^2 \leq G_g^2$ and $\|g_n - \mathbb{E}[g_n]\|_*^2 \leq \sigma_g^2$ for some finite constants G_g and σ_g.

Similarly, we consider properties of the predictive model Φ_n that is used to estimate the gradient of the next per-round loss. Let $\mathcal{P}$ denote the class of these models (i.e. $\Phi_n \in \mathcal{P}$), which can potentially be *stochastic*. We make assumptions on the size of $\hat{g}_n$ and its variance.

Assumption 10.G.2. $\|\mathbb{E}[\hat{g}_n]\|_*^2 \leq G_{\hat{g}}^2$ and $\mathbb{E}[\|\hat{g}_n - \mathbb{E}[\hat{g}_n]\|_*^2] \leq \sigma_{\hat{g}}^2$ for some finite constants $G_{\hat{g}}$ and $\sigma_{\hat{g}}$.

Additionally, we assume these models are Lipschitz continuous.

Assumption 10.G.3. There is a constant $L \in [0, \infty)$ such that, for any instantaneous cost ψ and any $\Phi \in \mathcal{P}$, it satisfies $\|\mathbb{E}[\Phi(\pi)] - \mathbb{E}[\Phi(\pi')]\|_* \leq L\|\pi - \pi'\|$.

Lastly, as PICCOLO is agnostic to the base algorithm, we assume the local norm $\|\cdot\|_n$ chosen by the base algorithm at round n satisfies $\|\cdot\|_n^2 \geq \alpha_n \|\cdot\|^2$ for some $\alpha_n > 0$. This condition implies that $\|\cdot\|_{n,*}^2 \leq \frac{1}{\alpha_n}\|\cdot\|_*^2$. In addition, we assume α_n is non-decreasing so that $M_N = O(\alpha_N)$ in Assumption 10.5.1, where the leading constant in the bound $O(\alpha_N)$ is proportional to $|\Pi|$, as commonly chosen in online convex optimization.

10.G.2 A Useful Lemma

We study the bound in Theorem 10.5.1 under the assumptions made in the previous section. We first derive a basic inequality, following the idea in (Cheng et al., 2019b, Lemma 4.3).

Lemma 10.G.1. *Under Assumptions 10.G.1, 10.G.2, and 10.G.3, it holds*

$$\mathbb{E}[\|e_n\|_{*,n}^2] = \mathbb{E}[\|g_n - \hat{g}_n\|_{*,n}^2] \leq \frac{4}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + L_n^2\|\pi_n - \hat{\pi}_n\|_n^2 + E_n(\Phi_n)\right)$$

where $E_n(\Phi_n) = \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n, \psi_n)]\|_*^2$ *is the prediction error of model* Φ_n*.*

Proof. Recall $\hat{g}_n = \Phi_n(\hat{\pi}_n, \psi_n)$. Using the triangular inequality, we can simply derive

$$\begin{aligned}
&\mathbb{E}[\|g_n - \hat{g}_n\|_{*,n}^2] \\
&\leq 4\left(\mathbb{E}[\|g_n - \mathbb{E}[g_n]\|_{*,n}^2] + \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n, \psi_n)]\|_{*,n}^2\right) \\
&\quad + 4\left(\|\mathbb{E}[\Phi_n(\pi_n, \psi_n)] - \mathbb{E}[\hat{g}_n]\|_{*,n}^2 + \mathbb{E}[\|\mathbb{E}[\hat{g}_n] - \hat{g}_n\|_{*,n}^2]\right) \\
&= 4\left(\mathbb{E}[\|g_n - \mathbb{E}[g_n]\|_{*,n}^2] + \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n, \psi_n)]\|_{*,n}^2\right) \\
&\quad 4\left(+\|\mathbb{E}[\Phi_n(\pi_n, \psi_n)] - \mathbb{E}[\Phi_n(\hat{\pi}_n, \psi_n)]\|_{*,n}^2 + \mathbb{E}[\|\mathbb{E}[\hat{g}_n] - \hat{g}_n\|_{*,n}^2]\right) \\
&\leq 4\left(\frac{1}{\alpha_n}\sigma_g^2 + \frac{1}{\alpha_n}E_n(\Phi_n) + \|\mathbb{E}[\Phi_n(\pi_n, \psi_n)] - \mathbb{E}[\Phi_n(\hat{\pi}_n, \psi_n)]\|_{*,n}^2 + \frac{1}{\alpha_n}\sigma_{\hat{g}}^2\right) \\
&\leq \frac{4}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + L^2\|\pi_n - \hat{\pi}_n\|_n^2 + E_n(\Phi_n)\right)
\end{aligned}$$

where the last inequality is due to Assumption 10.G.3. ■

10.G.3 Optimal Regret Bounds

We now analyze the regret bound in Theorem 10.5.1

$$\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi\rangle \leq M_N + \sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2 \tag{10.41}$$

We first gain some intuition about the size of

$$M_N + \mathbb{E}\left[\sum_{n=1}^{N} \frac{w_n^2}{2}\|e_n\|_{*,n}^2\right]. \tag{10.42}$$

Because when $\texttt{adapt}(h_n, H_{n-1}, e_n, w_n)$ is called in the Correction Step in (10.30) with the error gradient e_n as input, an optimal base algorithm (e.g. all the base algorithms listed in Section 10.C) would choose a local norm sequence $\|\cdot\|_n$ such that (10.42) is optimal. For example, suppose $\|e_n\|_*^2 = O(1)$ and $w_n = n^p$ for some $p > -1$. If the base algorithm is basic mirror descent (cf. Section 10.C), then $\alpha_n = O(\frac{w_{1:n}}{\sqrt{n}})$. By our assumption that $M_N = O(\alpha_N)$, it implies (10.42) can be upper bounded by

$$\begin{aligned} M_N + \mathbb{E}\left[\sum_{n=1}^N \frac{w_n^2}{2}\|e_n\|_{*,n}^2\right] &\le O\left(\frac{w_{1:N}}{\sqrt{N}}\right) + \left[\sum_{n=1}^N \frac{w_n^2\sqrt{n}}{2w_{1:n}}\|e_n\|_*^2\right] \\ &\le O\left(\frac{w_{1:N}}{\sqrt{N}} + \sum_{n=1}^N \frac{w_n^2\sqrt{n}}{w_{1:n}}\right) = O\left(N^{p+1/2}\right) \end{aligned}$$

which will lead to an optimal weighted average regret in $O(\frac{1}{\sqrt{N}})$.

PICCOLO actually has a better regret than the simplified case discussed above, because of the negative term $-\frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2$ in (10.41). To see its effects, we combine Lemma 10.G.1 with (10.41) to reveal some insights:

$$\mathbb{E}\left[\sum_{n=1}^N w_n \langle g_n, \pi_n - \pi\rangle\right] \tag{10.43}$$

$$\le O(\alpha_N) + \mathbb{E}\left[\sum_{n=1}^N \frac{w_n^2}{2}\|e_n\|_{*,n}^2 - \frac{1}{2}\|\pi_n - \hat{\pi}_n\|_{n-1}^2\right] \tag{10.44}$$

$$\le O(\alpha_N) + \mathbb{E}\left[\sum_{n=1}^N \frac{2w_n^2}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + L^2\|\pi_n - \hat{\pi}_n\|_n^2 + E_n(\Phi_n)\right) - \frac{\alpha_{n-1}}{2}\|\pi_n - \hat{\pi}_n\|^2\right]$$

$$= \left(O(\alpha_N) + \mathbb{E}\left[\sum_{n=1}^N \frac{2w_n^2}{\alpha_n}\left(\sigma_g^2 + \sigma_{\hat{g}}^2 + E_n(\Phi_n)\right)\right]\right) + \left(\mathbb{E}\left[\sum_{n=1}^N (\frac{2w_n^2}{\alpha_n}L^2 - \frac{\alpha_{n-1}}{2})\|\pi_n - \hat{\pi}_n\|^2\right]\right) \tag{10.45}$$

The first term in (10.45) plays the same role as (10.42); when the base algorithm has an optimal $\texttt{adapt}$ operation and $w_n = n^p$ for some $p > -1$, it would be in $O\left(N^{p+1/2}\right)$. Here we see that the constant factor in this bound is proportional to $\sigma_g^2 + \sigma_{\hat{g}}^2 + E_n(\Phi_n)$. Therefore, if the variances σ_g^2, $\sigma_{\hat{g}}^2$ of the gradients are small, the regret would mainly depend on the

prediction error $E_n(\Phi_n)$ of Φ_n. In the next section (Section 10.G.4), we will show that when Φ_n is learned online (as the authors in (Cheng et al., 2019b) suggest), on average the regret is close to the regret of using the best model in the hindsight. The second term in (10.45) contributes to $O(1)$ in the regret, when the base algorithm adapts properly to w_n. For example, when $\alpha_n = \Theta(\frac{w_{1:n}}{\sqrt{n}})$ and $w_n = n^p$ for some $p > -1$, then

$$\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n} L^2 - \frac{\alpha_{n-1}}{2} = \sum_{n=1}^{N} O(n^{p-1/2} - n^{p+1/2}) = O(1)$$

In addition, because $\|\pi_n - \hat{\pi}_n\|$ would converge to zero, the effects of the second term in (10.45) becomes even minor.

In summary, for a reasonable base algorithm and $w_n = n^p$ with $p > -1$, running PICCOLO has the regret bound, for any π,

$$\mathbb{E}\left[\sum_{n=1}^{N} w_n \langle g_n, \pi_n - \pi \rangle\right] = O(\alpha_N) + O\left(\frac{w_{1:N}}{\sqrt{N}}(\sigma_g^2 + \sigma_{\hat{g}}^2)\right) + O(1) + \mathbb{E}\left[\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n} E_n(\Phi_n)\right] \tag{10.46}$$

Suppose $\alpha_n = \Theta(|\Pi|\frac{w_{1:n}}{\sqrt{n}})$ and $w_n = n^p$ for some $p > -1$, This implies the inequality

$$\mathbb{E}\left[\sum_{n=1}^{N} \langle w_n g_n, \pi_n - \pi \rangle\right] \leq O(1) + C_{\Pi,\Phi} \frac{w_{1:N}}{\sqrt{N}} \tag{10.40}$$

where $C_{\Pi,\Phi} = O(|\Pi| + \sigma_g^2 + \sigma_{\hat{g}}^2 + \sup_n E_n(\Phi_n))$. The use of non-uniform weights can lead to a faster on average decay of the standing $O(1)$ term in the final weighted average regret bound, i.e.

$$\frac{1}{w_{1:N}} \mathbb{E}\left[\sum_{n=1}^{N} \langle w_n g_n, \pi_n - \pi \rangle\right] \leq O\left(\frac{1}{w_{1:N}}\right) + \frac{C_{\Pi,\Phi}}{\sqrt{N}}$$

In general, the authors in (Cheng et al., 2018a, 2019b) recommend using $p \ll N$ (e.g. in the range of $[0, 5]$) to remove the undesirable constant factor, yet without introducing large

multiplicative constant factor.

10.G.4 Model Learning

The regret bound in (10.46) reveals an important factor that is due to the prediction error $\mathbb{E}\left[\sum_{n=1}^{N} \frac{2w_n^2}{\alpha_n} E_n(\Phi_n)\right]$, where we recall $E_n(\Phi_n) = \|\mathbb{E}[g_n] - \mathbb{E}[\Phi_n(\pi_n)]\|_*^2$. Cheng et al. (2019b) show that, to minimize this error sum through model learning, a secondary online learning problem with per-round loss $E_n(\cdot)$ can be considered. Note that this is a standard weighted adversarial online learning problem (weighted by $\frac{2w_n^2}{\alpha_n}$), because $E_n(\cdot)$ is revealed after one commits to using model Φ_n.

While in implementation the exact function $E_n(\cdot)$ is unavailable (as it requires infinite data), we can adopt an unbiased upper bound. For example, Cheng et al. (2019b) show that $E_n(\cdot)$ can be upper bounded by the single- or multi-step prediction error of a transition dynamics model. More generally, we can learn a neural network to minimize the gradient prediction error directly. As long as this secondary online learning problem is solved by a no-regret algorithm, the error due to online model learning would contribute a term in $O(w_{1:N}\epsilon_{\mathcal{P},N}/\sqrt{N}) + o(w_{1:N}/\sqrt{N})$ in (10.46), where $\epsilon_{\mathcal{P},N}$ is the minimal error achieved by the best model in the model class $\mathcal{P}$ (see (Cheng et al., 2019b) for details).

10.H Experimental Details

10.H.1 Algorithms

Base Algorithms In the experiments, we consider three commonly used first-order online learning algorithms: ADAM, NATGRAD, and TRPO, all of which adapt the regularization online to alleviate the burden of learning rate tuning. We provide the decomposition of ADAM into the basic three operations in Section 10.C, and that of NATGRAD in Section 10.E. In particular, the adaptivity of NATGRAD is achieved by adjusting the step size based on a moving average of the dual norm of the gradient. TRPO adjusts the step size to minimize a given cost function (here it is a linear function defined by the first-order ora-

cle) within a pre-specified KL divergence centered at the current decision. While greedily changing the step size in every iteration makes TRPO an inappropriate candidate for adversarial online learning. Nonetheless, it can still be written in the form of mirror descent and allows a decomposition using the three basic operators; its `adapt` operator can be defined as the process of finding the maximal scalar step along the natural gradient direction such that the updated decision stays within the trust region. For all the algorithms, a decaying step size multiplier in the form $\eta/(1+\alpha\sqrt{n})$ is also used; for TRPO, it is used to specify the size of trust regions. The values chosen for the hyperparameters η and α can be found in Table 10.2. To the best of our knowledge, the conversion of these approaches into unbiased model-based algorithms is novel.

Reinforcement Learning Per-round Loss In iteration n, in order to compute the online gradient (10.5), GAE (Schulman et al., 2015a) is used to estimate the advantage function $A^{\pi_{n-1}}$. More concretely, this advantage estimate utilizes an estimate of value function $V^{\pi_{n-1}}$ (which we denote $\hat{V}_{\pi_{n-1}}$) and on-policy samples. We chosen $\lambda = 0.98$ in GAE to reduce influence of the error in $V^{\pi_{n-1}}$, which can be catastrophic. Importance sampling can be used to estimate $A^{\pi_{n-1}}$ in order to leverage data that are collected on-policy by running π_n. However, since we select a large λ, importance sampling can lead to vanishing importance weights, making the gradient extremely noisy. Therefore, in the experiments, importance sampling is not applied.

Gradient Computation and Control Variate The gradients are computed using likelihood-ratio trick and the associated advantage function estimates described above. A scalar control variate is further used to reduce the variance of the sampled gradient, which is set to the mean of the advantage estimates evaluated on newly collected data.

Policies and Value Networks Simple feed-forward neural networks are used to construct all of the function approximators (policy and value function) in the tasks. They have 1

hidden layer with 32 $\tanh$ units for all policy networks, and have 2 hidden layers with 64 $\tanh$ units for value function networks. Gaussian stochastic policies are considered, i.e., for any state $s \in \mathcal{S}$, $\pi(a|s)$ is Gaussian, and the mean of $\pi(a|s)$ is modeled by the policy network, whereas the diagonal covariance matrix is state independent (which is also learned). Initial value of $\log \sigma$ of the Gaussian policies -1.0, the standard deviation for initializing the output layer is 0.01, and the standard deviation for initialization hidden layer is 1.0. After the policy update, a new value function estimate $\hat{V}^{\pi_n}$ is computed by minimizing the mean of squared difference between $\hat{V}^{\pi_n}$ and $\hat{V}^{\pi_{n-1}} + \hat{A}^{\pi_n}$, where $\hat{A}^{\pi_n}$ is the GAE estimate using $\hat{V}^{\pi_{n-1}}$ and $\lambda = 0.98$, through ADAM with batch size 128, number of batches 2048, and learning rate 0.001. Value function is pretrained using examples collected by executing the randomly initialized policy.

Computing Model Gradients We compute $\hat{g}_n$ in two ways. The first approach is to use the simple heuristic that sets $\hat{g}_n = \Phi_n(\hat{\pi}_n)$, where Φ_n is some predictive models depending on the exact experimental setup. The second approach is to use the fixed-point formulation (10.9). This is realized by solving the equivalent optimization problem mentioned in the paper. In implementation, we only solves this problem approximately using some finite number of gradient steps; though this is insufficient to yield a stationary point as desired in the theory, we experimentally find that it is sufficient to yield improvement over the heuristic $\hat{g}_n = \Phi_n(\hat{\pi}_n)$.

Approximate Solution to Fixed-Point Problems of PICCOLO PICCOLO relies on the predicted gradient $\hat{g}_n$ in the Prediction Step. Recall ideally we wish to solve the fixed-point problem that finds h_n^* such that

$$h_n^* = \texttt{update}(\hat{h}_n, H_{n-1}, \Phi_n(\pi_n(h_n^*)), w_n) \tag{10.47}$$

and then apply $\hat{g}_n = \Phi_n(\pi_n(h_n^*))$ in the Prediction Step to get h_n, i.e.,

$$h_n = \texttt{update}(\hat{h}_n, H_{n-1}, \hat{g}_n, w_n)$$

Because h_n^* is the solution to the fixed-point problem, we have $h_n = h_n^*$. Such choice of $\hat{g}_n$ will fully leverage the information provided by Φ_n, as it does not induce additional linearization due to evaluating Φ_n at points different from h_n.

Exactly solving the fixed-point problem is difficult. In the experiments, we adopt a heuristic which computes an approximation to h_n^* as follows. We suppose $\Phi_n = \nabla f_n$ for some function f_n, which is the case e.g. when Φ_n is the simulated gradient based on some (biased) dynamics model. This restriction makes the fixed-point problem as finding a stationary point of the optimization problem $\min_{\pi \in \Pi} f_n(\pi) + B_{R_{n-1}}(\pi || \hat{\pi}_n)$. In implementation, we initialize the iterate in this subproblem as $\texttt{update}(\hat{h}_n, H_{n-1}, \Phi_n(\hat{\pi}_n), w_n)$, which is the output of the Prediction Step if we were to use $\hat{g}_n = \Phi_n(\hat{\pi}_n)$. We made this choice in initializing the subproblem, as we know that using $\hat{g}_n = \Phi_n(\hat{\pi}_n)$ in PICCOLO already works well (see the experiments) and it can be viewed as the solution to the fixed-point problem with respect to the linearized version of Φ_n at $\hat{\pi}_n$. Given the this initialization point, we proceed to compute the approximate solution to the fixed-point by applying the given base algorithm for 5 iterations and then return the last iterate as the approximate solution. For example, if the base algorithm is natural gradient descent, we fixed the Bregman divergence (i.e. its the Fisher information matrix as $\hat{\pi}_n$) and only updated the scalar stepsize adaptively along with the policy in solving this *regularized model-based RL problem* (i.e. $\min_{\pi \in \Pi} f_n(\pi) + B_{R_{n-1}}(\pi || \hat{\pi}_n)$). While such simple implementation is not ideal, we found it works in practice, though we acknowledge that a better implementation of the subproblem solver would improve the results.

10.H.2 Tasks

The robotic control tasks that are considered in the experiments are CartPole, Hopper, Snake, and Walker3D from OpenAI Gym (Brockman et al., 2016) with the DART physics engine (Lee et al., 2018a)[15]. CartPole is a classic control problem, and its goal is to keep a pole balanced in a upright posture, by only applying force to the cart. Hopper, Snake, and Walker3D are locomotion tasks, of which the goal is to control an agent to move forward as quickly as possible without falling down (for Hopper and Walker3D) or deviating too much from moving forward (for Snake). Hopper is monopedal and Walker3D is bipedal, and both of them are subjected to significant contact discontinuities that are hard or even impossible to predict.

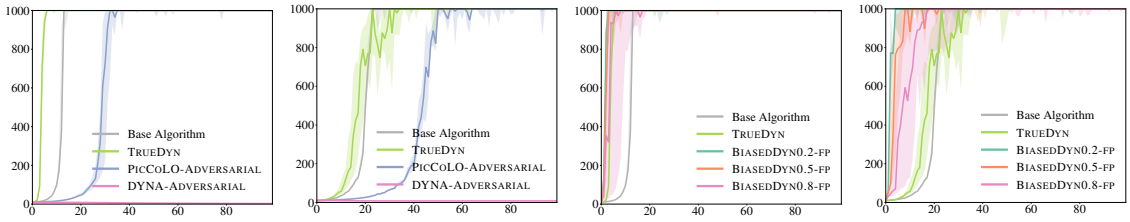
10.H.3 Full Experimental Results

In Figure 10.3, we empirically study the properties of PICCOLO that are predicted by theory on CartPole environment. In Figure 10.4, we "PICCOLO " three base algorithms: ADAM, NATGRAD, TRPO, and apply them on four simulated environments: Cartpole, Hopper, Snake, and Walker3D.

10.H.4 Experiment Hyperparameters

The hyperparameters used in the experiments and the basic attributes of the environments are detailed in Table 10.2.

[15]The environments are defined in DartEnv, hosted at https://github.com/DartEnv.



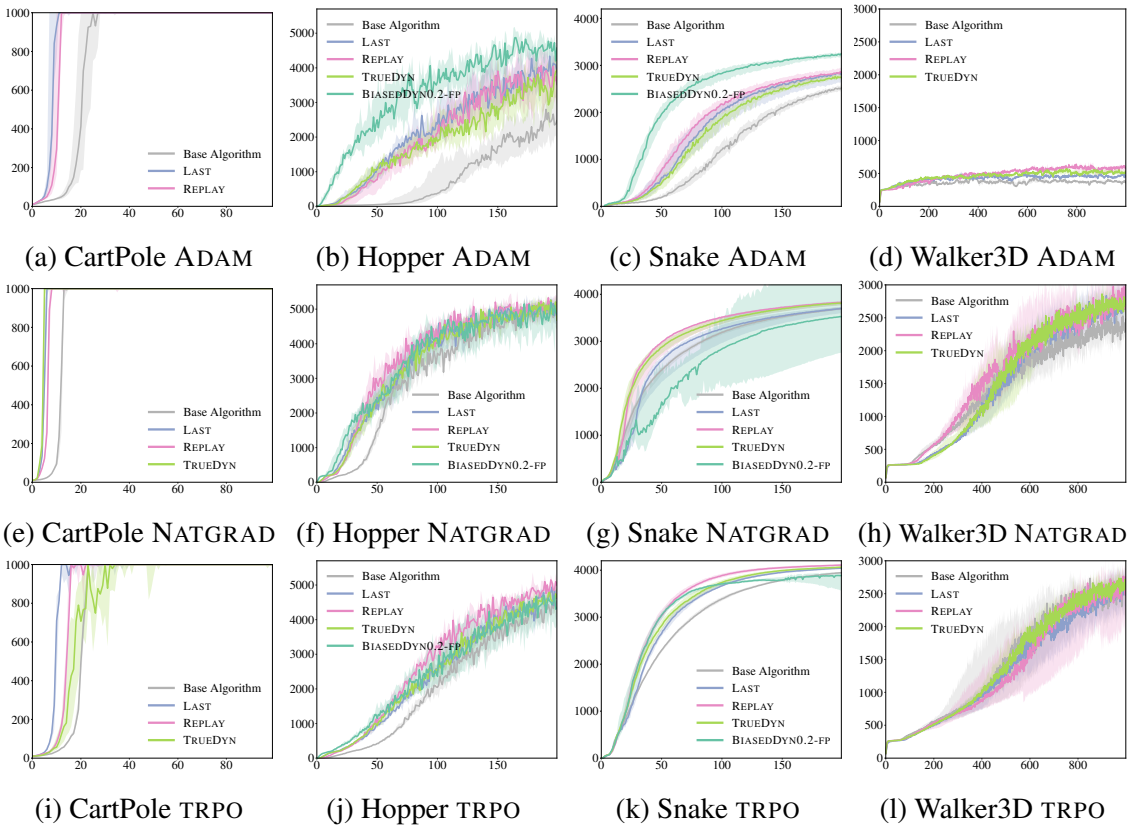
(a) Adv. model, NATGRAD (b) Adv. model, TRPO (c) Diff. fidelity, NATGRAD (d) Diff. fidelity, TRPO

Figure 10.3: Performance of PICCOLO with different predictive models on CartPole. x axis is iteration number and y axis is sum of rewards. The curves are the median among 8 runs with different seeds, and the shaded regions account for 25% percentile. The update rule, by default, is PICCOLO. For example TRUEDYN in (a) refers to PICCOLO with TRUEDYN predictive model. (a), (b): Comparison of PICCOLO and DYNA with adversarial model using NATGRAD and TRPO as base algorithms. (c), (d): PICCOLO with the fixed-point setting (10.9) with dynamics model in different fidelities. BIASEDDYN0.8 indicates that the mass of each individual robot link is either increased or decreased by 80% with probablity 0.5 respectively.

	CartPole	Hopper	Snake	Walker3D
Observation space dimension	4	11	17	41
Action space dimension	1	3	6	15
State space dimension	4	12	18	42
Number of samples from env. per iteration	4k	16k	16k	32k
Number of samples from model dyn. per iteration	4k	16k	16k	32k
Length of horizon	1,000	1,000	1,000	1,000
Number of iterations	100	200	200	1,000
Number of iterations of samples for REPLAY buffer	5	4	3	2 (3 for ADAM)
α [16]	0.1	0.1	0.1	0.01
η in ADAM	0.005	0.005	0.002	0.01
η in NATGRAD	0.05	0.05	0.2	0.2
η in TRPO	0.002	0.002	0.01	0.04

Table 10.2: Tasks specifics and hyperparameters.

[16] α and η appear in the decaying step size multiplier for all the algorithms in the form $\eta/(1+\alpha\sqrt{n})$. α influences how fast the step size decays. We chose α in the experiments based on the number of iterations.



(a) CartPole ADAM (b) Hopper ADAM (c) Snake ADAM (d) Walker3D ADAM

(e) CartPole NATGRAD (f) Hopper NATGRAD (g) Snake NATGRAD (h) Walker3D NATGRAD

(i) CartPole TRPO (j) Hopper TRPO (k) Snake TRPO (l) Walker3D TRPO

Figure 10.4: The performance of PICCOLO with different predictive models on various tasks, compared to base algorithms. The rows use ADAM, NATGRAD, and TRPO as the base algorithms, respectively. x axis is iteration number and y axis is sum of rewards. The curves are the median among 8 runs with different seeds, and the shaded regions account for 25% percentile.

Part III

Structral Policy Fusion

ABSTRACT

In the last part, we study another important question in policy optimization: the design of structural policies (Cheng et al., 2018b; Li et al., 2019b; Mukadam et al., 2019). Our goal here is to develop a rich policy class where every policy is provably stable. Such a policy class can supplement the statistical learning techniques developed in the first two parts of the thesis, so that they can be robustly and safely applied to robotics applications.

Toward this end, in Chapter 11, we develop a novel policy synthesis algorithm, called RMPflow, based on geometrically consistent transformations of Riemannian Motion Policies (RMPs). Given a set of RMPs designed for individual tasks, RMPflow can consistently combine them to generate a global policy, while simultaneously exploiting sparse structure for computational efficiency. Moreover, we show that this combined policy is Lyapunov stable, if each user-provided RMP belongs to the family of Geometric Dynamical Systems.

While RMPflow can be used as a framework for policy fusion, we can also treat it more generally as a differential computational graph for expressing motion policies with inherent stability. In Chapter 12, we formally establish this idea and propose a variation of RMPflow, called RMPfusion, which has extra weight functions to more finely control the policy fusion process of RMPflow.

Finally, in Chapter 13, we provide a general stability result of RMPflow based on Control Lyapunov Function (CLF), showing that RMPflow is capable of stably combining a large class of task-space policies that are stable with respect to a family of CLFs. Leveraging this finding, we then design an efficient CLF-based computational framework that can combine arbitrary subtask policies provided by users into a globally stable policy. In view of policy optimization, this result again shows that RMPflow is indeed a rich, Lyapunov-stable policy class, making RMPflow an ideal candidate for policy optimization with requirements of safety and interpretability.

NOTATION

Table 10.3: Notation for RMPflow

Symbol	Definition
$\mathcal{C}$	the configuration space
$\mathcal{T}$	the task space
$\mathbf{q}$	the coordinate of the configuration space
$\dot{\mathbf{q}}$	the velocity in view of $\mathbf{q}$
$\ddot{\mathbf{q}}$	the acceleration in view of $\mathbf{q}$
$\pi(\mathbf{q}, \dot{\mathbf{q}})$	the global motion policy
ψ	the transformation between manifolds
$\mathbf{J}$	the Jacobian matrix of the transformation
$(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$	the canonical form of an RMP on a manifold $\mathcal{M}$
$[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$	the natural form of the RMP above, where $\mathbf{f} = \mathbf{Ma}$
$\mathbf{a}$	the desired acceleration in the RMP
$\mathbf{f}$	the desired force in the RMP
$\mathbf{M}$	the virtual inertia in the RMP
$GDS(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$	the geometric dynamical system on a manifold $\mathcal{M}$
$\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})$	the metric matrix in the GDS
$\mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})$	the damping matrix the GDS
$\Phi(\mathbf{q})$	the potential function the GDS
$\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})$, $\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})$	the curvature terms derived from the metric $\mathbf{G}$
$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$	the Coriolis term in a simple mechanical system
$V(\mathbf{q}, \dot{\mathbf{q}})$	the global Lyapunov function
$T\mathcal{M}$	the tangent bundle of a manifold $\mathcal{M}$
$TT\mathcal{M}$	the double tangent bundle of a manifold $\mathcal{M}$

CHAPTER 11
A GEOMETRIC FRAMEWORK FOR POLICY FUSION

11.1 Introduction

In this chapter, we develop a new motion generation and control framework that enables globally stable controller design for *intrinsically* non-Euclidean spaces, i.e., spaces defined by non-constant Riemannian metrics with non-trivial curvature. Non-Euclidean geometries arise commonly in the natural world, in particular in the problem of obstacle avoidance. When obstacles are present, straight lines are no longer a reasonable definition of geodesics (namely, paths with shortest distance). Rather, geodesics must naturally flow around these obstacles that, in effect, become holes in the space and block trajectories from passing. This behavior implies a form of non-Euclidean geometry, because the space is naturally curved by the presence of obstacles.

The planning literature has made substantial progress in modeling non-Euclidean task-space behaviors, but at the expense of efficiency and reactivity. Starting with early differential geometric models of obstacle avoidance (Rimon and Koditschek, 1991) and building toward modern planning algorithms and optimization techniques (Gammell, Srinivasa, and Barfoot, 2015; Ivan et al., 2013; Karaman and Frazzoli, 2011; LaValle, 2006; Mukadam et al., 2018; Ratliff, Toussaint, and Schaal, 2015b; Toussaint, 2009; Watterson et al., 2018), these algorithms can calculate highly nonlinear trajectories. However, they are often computationally intensive, sensitive to noise, and unresponsive to perturbation. In addition, the internal nonlinearities of robots due to kinematic constraints are sometimes simplified in the optimization. While fast approximation and replanning heuristics have been proposed, the above characteristics in their nature make them unsuitable for motion generation in dynamic situations.

At the same time, a separate thread of literature, emphasizing fast reactive responses over computationally expensive planning, developed efficient closed-loop control techniques such as operational space control (Khatib, 1987). But while these techniques account for internal geometries from the robot's kinematic structure, they assume simple Euclidean geometry in task spaces (Peters et al., 2008; Udwadia, 2003), thus failing to provide a complete treatment of the external geometries. For example, the unified formulation of operational space control (Peters et al., 2008; Udwadia, 2003) is implicitly built on a classical mechanics concept called Gauss's principle of least constraint (Udwadia and Kalaba, 1996) which assumes each task space is Euclidean. As a result, obstacle avoidance, e.g., has to rely on *extrinsic* potential functions, leading to undesirable deceleration behavior when the robot is close to the obstacle. If somehow the non-Euclidean geometry can be *intrinsically* considered, then fast obstacle avoidance motion would naturally arise as traveling along the induced geodesic. The need for a holistic solution to motion generation and control has motivated a number of recent system architectures that tightly integrate planning and control (Kappler et al., 2018; Mukadam et al., 2017).

We improve upon these works by developing a new approach to synthesizing control policies that can accommodate and leverage the modeling capacity of intrinsically non-Euclidean robotics tasks. Taking inspiration from Geometric Control Theory (Bullo and Lewis, 2004),[1] we design a novel recursive algorithm, RMPflow, for representing nonlinear policies, based on a recently proposed mathematical object known as the Riemannian Motion Policy (RMP) (Ratliff, Issac, and Kappler, 2018). RMPflow enables geometrically consistent fusion of many component policies defined in non-Euclidean task spaces that are related through a tree structure. In essence, RMPflow computes a robot's acceleration by solving a high-dimensional weighted least-squared problem in which the weight matrices are nonlinear functions of the robot's position and velocity (i.e. the system's state). While solving a high-dimensional optimization problem seems computationally difficult at first

[1] See Section 11.6.1 for a discussion of why geometric mechanics and geometric control theory constitute a good starting point.

glance, RMPflow avoids this pitfall by computing the policy through performing forward and backward message passing along the tree structure that relates different task spaces. As a result, the computation paths shared across different tasks can be leveraged to achieve efficiency. Algorithmically, we can view RMPflow as mimicking the Recursive Newton-Euler algorithm (Walker and Orin, 1982) in structure, but generalizing it beyond rigid-body systems to a broader class of highly nonlinear transformations and spaces.

In contrast to existing frameworks, our framework, through the use of nonlinear weight matrix functions, naturally models non-Euclidean task spaces with Riemannian metrics that are not only configuration dependent, but also *velocity* dependent. This allows RMPflow to consider, e.g., the *direction* a robot travels to define the importance weights in combing policies. For example, an obstacle, despite being close to the robot, can usually be ignored if robot is heading away from it. This new class of policies leads to an extension of Geometric Control Theory, building on a new class of non-physical mechanical systems we call geometric dynamical systems (GDS).

While RMPflow offers extra flexibility in control design, one might naturally ask if it is even stable, as the use of weight function introduces additional feedback signals that could destroy the original stability of the component policies. The answer to this question is affirmative. We prove that RMPflow is Lyapunov-stable. Moreover, we show that the construction of RMPflow is coordinate-free. In particular, when using RMPflow, robots can be viewed each as different parameterizations of the same task space, defining a precise notion of behavioral consistency between robots. Additionally, under this framework, the implicit curvature arising from non-constant Riemannian metrics (which may be roughly viewed as configuration-velocity dependent inertia matrices in operational space control) produces nontrivial and intuitive policy contributions that are critical to guaranteeing stability and generalization across embodiments.

We demonstrate the properties of RMPflow in simulations and experiments. Our experimental results illustrate how these curvature terms can be impactful in practice, generating

nonlinear geodesics that result in curving or orbiting around obstacles. Furthermore, we demonstrate the utility of our framework with a fully reactive real-world system implementation on multiple dual-arm manipulation problems.

This chapter is based on our paper (Cheng et al., 2019e). An earlier version of this chapter was published as (Cheng et al., 2018b) with more details in a corresponding technical report (Cheng et al., 2018c) which includes many specific examples of the RMPs used in the experiments (Appendix D). The design of RMPflow is highly inspired by the seminal work of RMPs (Ratliff, Issac, and Kappler, 2018) that promotes the concept of including geometric information in policy fusion. This chapter and its former version (Cheng et al., 2018b) formalize the original intuition (Ratliff, Issac, and Kappler, 2018) and further extend this idea to geometric mechanics and beyond. Increasingly RMPs and RMPflow have been applied broadly into robotic systems, finding applications in autonomous navigation (Meng et al., 2019), manipulation systems (Kappler et al., 2018), reactive logical task sequencing (Paxton et al., 2019), tactile servoing (Sutanto et al., 2019), and multi-agent systems (Li et al., 2019a,b).

11.2 Motion Generation and Control

Motion generation and control can be formulated as the problem of transforming curves from the configuration space $\mathcal{C}$ to the task space $\mathcal{T}$. Specifically, let the configuration space $\mathcal{C}$ be a d-dimensional smooth manifold. A robot's motion can be described as a curve $q : [0, \infty) \to \mathcal{C}$ such that the robot's configuration at time t is a point $q(t) \in \mathcal{C}$. Without loss of generality, suppose $\mathcal{C}$ has a global coordinate $\mathbf{q} : \mathcal{C} \to \mathbb{R}^d$, called the *generalized coordinate*; for brevity, we identify the curve q with its coordinate expression $\mathbf{q} \circ q$ and write $\mathbf{q}(q(t))$ as $\mathbf{q}(t) \in \mathbb{R}^d$. A typical example of the generalized coordinate is the joint angles of a d-DOF (degrees-of-freedom) robot: we denote $\mathbf{q}(t)$ as the joint angles at time t and $\dot{\mathbf{q}}(t)$, $\ddot{\mathbf{q}}(t)$ as the joint velocities and accelerations, respectively. To describe the tasks, we consider another manifold $\mathcal{T}$, the task space, which is related to the configuration space

$\mathcal{C}$ through a smooth *task map* $\psi : \mathcal{C} \to \mathcal{T}$. The task space $\mathcal{T}$ can be the end-effector position/orientation (Albu-Schaffer and Hirzinger, 2002; Khatib, 1987), or more generally can be a space that describes whole-body robot motion, e.g., in simultaneous tracking and collision avoidance (Lo, Cheng, and Huang, 2016; Sentis and Khatib, 2006). Under this setup, thus the goal of motion generation and control can be viewed as designing the curve q (in a closed-loop manner) so that the transformed curve $\psi \circ q$ exhibits desired behaviors on the task space $\mathcal{T}$.

To simplify the exposition, below we suppose that the robot's dynamics have been feedback linearized and restrict our attention to designing acceleration-based controllers. We remark that a torque-based setup can be similarly derived by redefining the pseudo-inverse in `resolve` in Section 11.4.4 in terms of the inner product space induced by the robot's physical inertia on $\mathcal{C}$ (Peters et al., 2008), so long as the system is fully actuated and the inverse dynamics can be modeled.

11.2.1 Notation

For clarity, we use boldface to distinguish the coordinate-dependent representations from abstract objects; e.g. we write $q(t) \in \mathcal{C}$ and $\mathbf{q}(t) \in \mathbb{R}^d$. In addition, we will often omit the time- and input-dependency of objects unless necessary; e.g., we may write $q \in \mathcal{C}$ and $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$. For derivatives, we use both symbols ∇ and ∂, with a transpose relationship: for $\mathbf{x} \in \mathbb{R}^m$ and a differential map $\mathbf{y} : \mathbb{R}^m \to \mathbb{R}^n$, we write $\nabla_{\mathbf{x}} \mathbf{y}(\mathbf{x}) = \partial_{\mathbf{x}} \mathbf{y}(\mathbf{x})^\top \in \mathbb{R}^{m \times n}$. This choice of notation allows us to write $\nabla_{\mathbf{y}} f(\mathbf{y}) \in \mathbb{R}^n$ when f is a scalar function and perform chain-rule $\partial_x f(\mathbf{y}(\mathbf{x})) = \partial_{\mathbf{y}} f(\mathbf{y}) \partial_{\mathbf{x}} \mathbf{y}(\mathbf{x})$ in the usual way. For a matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$, we denote $\mathbf{b}_i = (\mathbf{M})_i$ as its ith column and $M_{ij} = (\mathbf{M})_{ij}$ as its (i, j) element. To compose a matrix from vector or scalar elements, we use $(\cdot)^:$ for vertical (or matrix) concatenation and $[\cdot]^:$ for horizontal concatenation. For example, we write $\mathbf{M} = [\mathbf{b}_i]_{i=1}^m = (M_{ij})_{i,j=1}^m$ and $\mathbf{M}^\top = (\mathbf{b}_i^\top)_{i=1}^m = (M_{ji})_{i,j=1}^m$. We use $\mathbb{R}_+^{m \times m}$ and $\mathbb{R}_{++}^{m \times m}$ to denote the symmetric, positive semi-definite/definite matrices, respectively.

11.2.2 Motion Policies and the Geometry of Motion

We model motion as a second-order differential equation of $\ddot{\mathbf{q}} = \pi(\mathbf{q}, \dot{\mathbf{q}})$, where we call π a *motion policy* and $(\mathbf{q}, \dot{\mathbf{q}})$ the *state*. In contrast to an open-loop trajectory, which forms the basis of many motion planners, a motion policy expresses the entire continuous collection of its integral trajectories and therefore is robust to perturbations. Motion policies can model many adaptive behaviors, such as reactive obstacle avoidance (Erez et al., 2013; Kappler et al., 2018) or responses driven by planned Q-functions (Todorov, 2006), and their second-order formulation enables rich behavior that cannot be realized by the velocity-based approach (Liegeois, 1977).

The geometry of motion has been considered by many planning and control algorithms. Geometrical modeling of task spaces is used in topological motion planning (Ivan et al., 2013), and motion optimization has leveraged Hessian to exploit the natural geometry of costs (Dong et al., 2016; Mukadam, Yan, and Boots, 2016; Ratliff et al., 2009; Toussaint, 2009). Ratliff et al. (Ratliff, Toussaint, and Schaal, 2015b), e.g., use the workspace geometry inside a Gauss-Newton optimizer and generate natural obstacle-avoiding reaching motion through traveling along geodesics of curved spaces.

Geometry-aware motion policies were also developed in parallel in controls. Operational space control is the best example (Khatib, 1987). Unlike the planning approaches, operational space control focuses on the internal geometry of the robot and considers only simple task-space geometry: it reshapes the workspace dynamics into a simple spring-mass-damper system with a constant inertia matrix, enforcing a form of Euclidean geometry in the task space. By contrast, pure potential-field approaches (Flacco et al., 2012; Kaldestad et al., 2014; Khatib, 1985) fail to realize this idea of task-space geometry and lead to inconsistent behaviors across robots. Variants of operational space control have been proposed to consider different metrics (Lo, Cheng, and Huang, 2016; Nakanishi et al., 2008; Peters et al., 2008), task hierarchies (Platt, Abdallah, and Wampler, 2011; Sentis and Khatib, 2006), and non-stationary inputs (Ijspeert et al., 2013).

While these algorithms have led to many advances, we argue that their isolated focus on either the internal or the external geometry limits the performance. The planning approach fails to consider reactive dynamic behavior; the control approach cannot[2] model the effects of velocity dependent metrics, which are critical to generating sensible obstacle avoidance motions, as discussed in the introduction. While the benefits of velocity dependent metrics was recently explored using RMPs (Ratliff, Issac, and Kappler, 2018), a systematic understanding of its properties, like stability, is still an open question.

11.3 From Operational Space Control to Geometric Control

We set the stage for our development of RMPflow and geometric dynamical systems (GDSs) in Section 11.4 and 11.5 by first giving some background on the key tools central to this work. Specifically, we give a tutorial on the controller design technique known as energy shaping and the geometric formulation of classical mechanics, both of which are commonly less familiar to robotics researchers. Then we will show that geometric control (Bullo and Lewis, 2004), which to a great extent developed independently of operational space control within a distinct community, nicely summarizes these two ideas and leads to techniques of leveraging energy shaping within the context of geometric mechanics.

This section targets at readers more familiar with operational space control and introduces many of the relevant ideas in a way that we hope is more accessible than the traditional exposition which assumes a background in differential geometry. We begin with a review of classical operational space control wherein tasks are represented as hard constraints on the mechanical system, and then show how energy shaping and the geometric mechanics formalism enable us to easily develop provably stable operational space controllers that simultaneously trade off many tasks. The material presented in this section primarily rehashes existing techniques from a perhaps unfamiliar community, restating them in a way that should be more natural to researchers familar with operational space con-

[2]Existing works, like variants of operational space control and designs centered around Geometric Control Theory (Bullo and Lewis, 2004), can consider at most position-dependent metrics.

trol. We end with a discussion of the limitations of these geometric control techniques that RMPflow and GDSs will address in Section 11.4 and 11.5.

11.3.1 Energy Shaping and Classical Operational Space Control

Energy shaping is a controller design technique, wherein the designer first configures a virtual mechanical system by shaping its kinetic and potential energies to exhibit a certain behavior, and then drive the robot's dynamics to mimic that virtual system. This scheme overall generates a control law with a well-defined Lyapunov function, given as the virtual system's total energy, and therefore has provable stability.

For instance, the earliest form of operational space control (Khatib, 1987) formulates a virtual system that places all mass at the end-effector. Behavior is then shaped by applying potential energy functions (regulated by a damper) to that virtual mass (e.g. by connecting the end-effector to a target using a virtual (damped) spring). Controlling the system to behave like that virtual system then generates a control law whose stability is governed by the total energy of that virtual point-mass system. In this context, the choice of virtual mechanical system (the point end-effector mass) is a form of *kinetic energy shaping*, and the subsequent choice of potential energy applied to that point end-effector mass is known as *potential energy shaping*. This particular pattern of task-centric kinetic and potential energy shaping, is common throughout the operational space control literature.

A similar theme can be found in (Peters et al., 2008). Here the virtual mechanical systems are designed by constraining an existing mechanical system (e.g. the robot's original dynamics) to satisfy task constraints. This is achieved by designing controllers around a generalized form of Gauss's principle of least constraint (Udwadia and Kalaba, 1996), so that virtual mechanical systems would behave in a sense as similarly as possible to the true robotic mechanical system while realizing the required task accelerations. In other words, the energies of the original mechanical system are reshaped to that given by the task constraints.

In essence, these early examples above are based on the idea that faithful execution of the task enables a simplified stability analysis as long as the task space behavior is itself well-understood and stable. This style of simplified analysis and the controller design has been successful in practice. Nonetheless, it imposes a limitation that the controllers cannot have more tasks than the number of DOF in the system. This becomes particularly problematic when one wishes to introduce more complex auxiliary behaviors, such as collision avoidance where the number of tasks might scale with the number of obstacles and the number of control points on the robot's body.

The rest of this section is dedicated to unify and then generalize these ideas through the lens of geometric mechanics, so that we can use operational space control to handle these more complex settings of many competing tasks, by using nuanced weighted priorities that might change as a function of the robot's configuration. However, we will eventually see that even this is still not quite sufficient for representing many common behaviors (see Section 11.3.5). The insights into sources of these limitation learned in this section are the motivation of our more in-depth subsequent development of RMPflow and geometric dynamical systems (GDSs).

11.3.2 A Simple First Step toward Weighted Priorities

This section leverages Gauss's principle of least constraint (different from the techniques mentioned briefly in Section 11.3.1 (Peters et al., 2008)) to illustrate the concept of energy shaping, which will be used more abstractly below to derive a simple technique for combining multiple task-space policies.

Gauss's Principle

Gauss's principle of least constraint states that a nonlinearly constrained collection of particles evolves in a way that is most similar to its unconstrained evolution, as long as this notion of similarity is measured using the inertia-weighted squared error (Udwadia,

2003). For example, let us consider N particles: $\mathbf{x}_i \in \mathbb{R}^3$ with respective (positive) inertia $m_i \in \mathbb{R}_+$, for $i = 1, \ldots, N$. Then the acceleration $\ddot{\mathbf{x}}_i$ of the i^{th} particle under Gauss's principle can be written as

$$\ddot{\mathbf{x}} = \arg\min_{\ddot{\mathbf{x}}' \in \mathcal{A}} \frac{1}{2} \|\ddot{\mathbf{x}}^d - \ddot{\mathbf{x}}'\|^2_{\mathbf{M}} \tag{11.1}$$

where $\mathcal{A}$ denotes the set of admissible constrained accelerations. To simplify the notation, we've stacked[3] the particle accelerations into a vector $\ddot{\mathbf{x}} = (\ddot{\mathbf{x}}_1; \ldots; \ddot{\mathbf{x}}_N)$ and construct a diagonal matrix $\mathbf{M} = \text{diag}(m_1\mathbf{I}, \ldots, m_N\mathbf{I})$, where $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix.

Kinematic Control-Point Design

Let us use the above idea to design a robot controller. If we define many kinematic control points $\mathbf{x}_i \in \mathbb{R}^3, i = 1, \ldots, N$ distributed across the robot's body and calculate a desired acceleration at those points $\ddot{\mathbf{x}}_i^d$, a sensible way to trade off these different accelerations is through the following quadratic program (QP):

$$\min_{\ddot{\mathbf{x}}_i} \sum_{i=1}^{N} \frac{m_i}{2} \|\ddot{\mathbf{x}}_i^d - \ddot{\mathbf{x}}_i\|^2 \quad \text{s.t. } \ddot{\mathbf{x}}_i = \mathbf{J}_i\ddot{\mathbf{q}} + \dot{\mathbf{J}}_i\dot{\mathbf{q}}, \tag{11.2}$$

where $m_i > 0$ is the importance weight, $\mathbf{x}_i = \psi_i(\mathbf{q})$ is the forward kinematics map to the i^{th} control point and $\mathbf{J}_i = \partial_{\mathbf{q}_i}\psi_i$ is its Jacobian. This QP states that the system should follow the desired accelerations as well as possible, with (constant) tradeoff priorities m_i in the event the tasks cannot be achieved exactly, subject to the physical kinematic constraints on how each control point can accelerate.[4]

Comparing this QP to that given by Gauss's principle in (11.1), one can immediately see that its solution gives the constrained dynamics of a mechanical system defined by N point particles of mass m_i with unconstrained accelerations $\ddot{\mathbf{x}}_i^d$ and acceleration con-

[3] We use the notation $\boldsymbol{v} = (\boldsymbol{v}_1; \boldsymbol{v}_2; , \ldots, \boldsymbol{v}_N)$ to denote stacking of vectors $\boldsymbol{v}_i \in \mathbb{R}^3$ into a single vector $\boldsymbol{v} \in \mathbb{R}^{3N}$.

[4] As mentioned we assumed the system has been feedback linearized so we focus on acceleration only.

straints $\ddot{\mathbf{x}}_i = \mathbf{J}_i\ddot{\mathbf{q}} + \dot{\mathbf{J}}_i\dot{\mathbf{q}}$. In particular, if $\ddot{\mathbf{x}}_i^d = -m_i^{-1}\nabla\phi_i - \beta_i\dot{\mathbf{x}}_i$ for some non-negative potential function ϕ_i and constant β_i, we arrive at a mechanical system with total energy $\sum_{i=1}^{N}\left(\frac{m_i}{2}\|\dot{\mathbf{x}}_i\|^2 + \phi_i(\mathbf{x}_i)\right)$. Controlling the robot system according to desired accelerations $\ddot{\mathbf{q}}^*$ given by solving (11.2) ensures that this total energy dissipates at a rate defined by the collective non-negative dissipation terms $\sum_i m_i\beta_i\|\dot{\mathbf{x}}_i\|^2$. This total energy, therefore, acts as a Lyapunov function.

This kinematic control-point design technique utilizes now more explicitly the methodology of energy shaping. In this case, we use Gauss's principle to design a virtual mechanical system that strategically distributes point masses throughout the robot's body at key control points (kinetic energy shaping). We then apply (damped) virtual potential functions to those masses to generate behavior (potential energy shaping). In combination, we see that the resulting system can be viewed as a QP which tries to achieve all tasks simultaneously as well as it can. Since exact replication of all tasks is impossible, the QP uses the mass values as relative priorities to define how the system should trade off task errors when necessary.

11.3.3 Abstract Task Spaces: Simplified Geometric Mechanics

The controller we just described demonstrates the core concept around energy shaping, but is limited by requiring that tasks be designed specifically on kinematic control-points distributed *physically* across the robot's body. Usually task spaces are often more abstract than that, and most generally we consider any task space that can be described as a nonlinear map from the configuration space.

This abstraction is common in trajectory optimization. For instance, (Toussaint, 2009) describes some abstract topological spaces for behavior creation which enable behaviors such as wrapping an arm around a pole and unwrapping it, and abstract models of workspace geometry are represented in (Mainprice, Ratliff, and Schaal, 2016; Ratliff, Toussaint, and Schaal, 2015b) by designing high-dimensional task spaces consisting of stacked (proximity

weighted) local coordinate representations of surrounding obstacles conveying how obstacles shape the space around them. Likewise, similar abstract spaces are highly relevant for describing common objectives in operational space control problems. Specifically, spaces of interest include one-dimensional spaces encoding distances to barrier constraints such as joint limits and obstacles, distances to targets, spaces of quaternions, and the joint space itself; all of these are more abstract than specific kinematic control-points. In order to generalize these ideas to abstract task spaces we need better tools. Below we show the insights from geometric mechanics and geometric control theory (Bullo and Lewis, 2004) provide the generalization that we need.

Quick Review of Lagrangian Mechanics

Lagrangian mechanics is a reformulation of classical mechanics that derives the equations of motion by applying the Euler-Lagrange equation the Lagrangian of the mechanical system (Taylor, 2005). Specifically, given a generalized inertia matrix $\mathbf{M}(\mathbf{q})$ and a potential function $\Phi(\mathbf{q})$, the Lagrangian is the difference between kinetic and potential energies:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q})\dot{\mathbf{q}} - \Phi(\mathbf{q}). \tag{11.3}$$

The Euler-Lagrange equation is given by

$$\frac{d}{dt}\partial_{\dot{\mathbf{q}}}\mathcal{L} - \partial_{\mathbf{q}}\mathcal{L} = \tau_{\text{ext}} \tag{11.4}$$

where τ_{ext} is the external force applied on the system. Applying (11.4) to the Lagrangian (11.3) gives the equations of motion,

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \nabla\Phi(\mathbf{q}) = \tau_{\text{ext}}, \tag{11.5}$$

where $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \dot{\mathbf{M}}\dot{\mathbf{q}} - \frac{d}{dt}\left(\frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}\right)$. For convenience, we will define this term as

$$\boldsymbol{\xi}_{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) = \dot{\mathbf{M}}\dot{\mathbf{q}} - \frac{d}{dt}\left(\frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}\right) \tag{11.6}$$

which will play an important role when we discuss about the geometry of implicit task spaces. (This definition is consistent with the curvature term in GDSs that we later generalize.)

Ambient Geometric Mechanics

Geometric mechanics (Bullo and Lewis, 2004) is a reformulation of classical mechanics that builds on the observation that classical mechanical systems evolve as geodesics across a Riemannian manifold whose geometry is defined by the system's inertia matrix. We can see what this means by examining a simple example, which we will use to derive an operational space QP similar in form to (11.2).

Let $\mathbf{x} = \psi(\mathbf{q})$ be an arbitrary differentiable task map $\psi : \mathbb{R}^d \to \mathbb{R}^n$ where $n \geq d$. In practice, the full task map often consists of many smaller task maps stacked on top of one another, like a kinematic tree. We can define a positive definite matrix that changes as a function of configuration using $\mathbf{M}(\mathbf{q}) = \mathbf{J}(\mathbf{q})^\top \mathbf{J}(\mathbf{q})$ where $\mathbf{J}(\mathbf{q}) = \partial_{\mathbf{q}}\phi(\mathbf{q})$ is the Jacobian of the task map. Geometric mechanics states that we can think of $\mathbf{M}(\mathbf{q})$ as both the generalized inertia matrix of a mechanical system defining a dynamic behavior $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \tau_{\text{ext}}$ (see also (11.5)), and equivalently as a Riemannian metric defining an inner product $\langle \dot{\mathbf{q}}_1, \dot{\mathbf{q}}_2 \rangle_{\mathbf{M}} = \dot{\mathbf{q}}_1^\top \mathbf{M} \dot{\mathbf{q}}_2$ on the tangent space (for our purposes, the space of velocities $\dot{\mathbf{q}}$ at a given $\mathbf{q}$) of the configuration space $\mathcal{C}$ (the manifold where $\mathbf{q}$ lives). Note that since $\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$, the columns of $\mathbf{J}(\mathbf{q})$ span this tangent space, which we see from the dependency on $\mathbf{q}$ in $\mathbf{J}$ can change direction at different $\mathbf{q}$.

Because we can suppose $n \geq d$ in general[5], the set $\mathcal{X} = \{\mathbf{x} : \mathbf{x} = \psi(\mathbf{q}) \text{ for some } \mathbf{q} \in \mathcal{C}\}$

[5]This holds when ψ is full rank. Reduced rank ψ result in a similar geometry, but we would need to slightly modify the linear algebra used in the following discussion.

sweeps out a d-dimensional sub-manifold of the n-dimensional ambient Euclidean task space. In light of this picture, we can also view the tangent space as a first-order Taylor approximation to the surface at a point $\mathbf{x}_0 = \phi(\mathbf{q}_0)$ for some $\mathbf{q}_0$ in the sense $\mathbf{x} \approx \mathbf{x}_0 + \mathbf{J}(\mathbf{q})(\mathbf{q} - \mathbf{q}_0)$.

Indeed, one connection between the mechanical system and this geometry is clear: the kinetic energy of the mechanical system is given by the norm of $\dot{\mathbf{q}}$ with respect to the inner product defined by the metric $\mathbf{M}(\mathbf{q})$:

$$\mathcal{K}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q})\dot{\mathbf{q}} = \|\dot{\mathbf{q}}\|^2_{\mathbf{M}} = \langle \dot{\mathbf{q}}, \dot{\mathbf{q}} \rangle_{\mathbf{M}}. \tag{11.7}$$

Likewise, in this particular case, since $\mathbf{M} = \mathbf{J}^\top \mathbf{J}$ we see that specifically the kinetic energy is given by the Euclidean velocity through the task space

$$\mathcal{K}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q})\dot{\mathbf{q}} = \frac{1}{2}\dot{\mathbf{q}}^\top \big(\mathbf{J}^\top \mathbf{J}\big)\dot{\mathbf{q}} = \frac{1}{2}\|\dot{\mathbf{x}}\|^2. \tag{11.8}$$

More generally, for the same reason, Euclidean inner products between velocities in the task space induce these generalized inner products in the configuration space in the sense $\dot{\mathbf{x}}_1^\top \dot{\mathbf{x}}_2 = \dot{\mathbf{q}}_1^\top \mathbf{M} \dot{\mathbf{q}}_2$. This connection between task space and configuration space inner products, exemplified by the equivalence between task space velocity and the system's kinetic energy offers a concrete connection between mechanics and geometry, and we can exploit to link the system's equations of motion to geodesics across $\mathcal{X}$.

For systems without potential functions and external forces, we can get some insight into the connection between dynamics and geodesics from the view point of Lagrangian mechanics (reviewed in Section 11.3.3) as well. The Lagrangian (11.3) in this case simplifies to $\mathcal{L} = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}\dot{\mathbf{q}} - \Phi(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}\dot{\mathbf{q}}$. The Euler-Lagrange equation in (11.4) is the first-order optimality condition of an *action functional* which measures the time integral of

the Lagrangian across a trajectory. In this case, it takes on a nice minimization form

$$\min_{\xi} \int_a^b \frac{1}{2} \dot{\mathbf{q}}^\top \mathbf{M} \dot{\mathbf{q}} dt \;\; \Leftrightarrow \;\; \min_{\xi} \int_a^b \frac{1}{2} \|\dot{\mathbf{x}}\|^2 dt, \tag{11.9}$$

where ξ is a trajectory through the configuration space $\mathcal{C}$. One can show that these trajectories are length-minimizing trajectories (i.e. solutions extremize the length functional $\int_a^b \frac{1}{2}\|\dot{\mathbf{x}}\| dt$), but with the additional property that the trajectories are of constant velocity.

This means the dyanmical system will curve across the manifold $\mathcal{X}$ along a trajectory that is as straight as possible without speeding up or slowing down. Another way to characterize that statement, is to say the system never accelerates tangentially to the sub-manifold $\mathcal{X}$, i.e. it has no component of acceleration parallel to the tangent space. The curve certainly must accelerate to avoid diverging from the sub-manifold $\mathcal{X}$, but that acceleration is always purely orthogonal to the tangent space. Since we know that $\mathbf{J}$ spans the tangent space, we can capture that sentiment fully in the following simple equation:

$$\mathbf{J}^\top \ddot{\mathbf{x}} = 0. \tag{11.10}$$

Plugging in $\ddot{\mathbf{x}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$ we get

$$\begin{aligned} \mathbf{J}^\top \ddot{\mathbf{x}} &= \mathbf{J}^\top \left(\mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} \right) = 0 \\ \Rightarrow \left(\mathbf{J}^\top \mathbf{J} \right) \ddot{\mathbf{q}} &+ \mathbf{J}^\top \dot{\mathbf{J}} \dot{\mathbf{q}} = 0. \end{aligned} \tag{11.11}$$

Comparing to (11.5) (with zero potential and external forces), since we already know $\mathbf{J}^\top \mathbf{J} = \mathbf{M}$, we can formally prove the connection between geodesics and classical mechanical dynamics if we can show that $\mathbf{J}^\top \dot{\mathbf{J}} \dot{\mathbf{q}} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$. The required calculation is fairly involved, so we omit it here but note for those inclined that it's easiest to perform using tensor notation and the Einstein summation convention as is common in differential geometry. This equivalence also appears as by-product fo our RMPflow and GDS analysis,

as we will revisit in Section 11.6.1 as Lemma 11.6.1.

Forced Mechanical Systems and Geometric Control

So far we have derived only the unforced behavior of this system as natural geodesic flow across the sub-manifold. To understand how desired accelerations contribute to the least squares properties of the system we express (11.11) in $\mathbf{x}$ by pushing them through the identity $\ddot{\mathbf{x}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$ and examine how arbitrary motion across the sub-manifold $\mathcal{X}$ decomposes. First, the equations of motion in $\ddot{\mathbf{x}}$ with (11.11), we get

$$\begin{aligned}\ddot{\mathbf{x}} &= \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} = -\mathbf{J}\left(\mathbf{J}^\top\mathbf{J}\right)^{-1}\mathbf{J}^\top\dot{\mathbf{J}}\dot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} \\ &= \left(\mathbf{I} - \mathbf{J}\left(\mathbf{J}^\top\mathbf{J}\right)^{-1}\mathbf{J}^\top\right)\dot{\mathbf{J}}\dot{\mathbf{q}} \\ &= \mathbf{P}_\perp\dot{\mathbf{J}}\dot{\mathbf{q}}, \end{aligned} \tag{11.12}$$

where in the final expression, the matrix $\mathbf{P}_\perp = \mathbf{I} - \mathbf{P}_{/\!/}$ with $\mathbf{P}_{/\!/} = \mathbf{J}\left(\mathbf{J}^\top\mathbf{J}\right)^{-1}\mathbf{J}^\top$ is the nullspace projection operator projecting onto the space orthogonal to the tangent space (spanned by the Jacobian $\mathbf{J}$). Note again these projections $\mathbf{P}_\perp$ and $\mathbf{P}_{/\!/}$ are functions of the configuration $\mathbf{q}$.

By construction, these geodesics accelerate only orthogonally to the tangent space. This implies that any trajectory, traveling on the sub-manifold $\mathcal{X}$ but deviating from geodesics, would necessarily maintain an acceleration component parallel to the tangent space, which we might write as $\ddot{\mathbf{x}}_{/\!/}$. Importantly, any such trajectory must still accelerate exactly as (11.12) in the orthogonal direction in order to stay moving along the sub-manifold $\mathcal{X}$. Therefore, we see that the overall acceleration of a trajectory on $\mathcal{X}$ can be decomposed nicely into the geodesic acceleration and the tangential acceleration:

$$\ddot{\mathbf{x}} = \mathbf{P}_\perp\dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{P}_{/\!/}\ddot{\mathbf{x}}_d, \tag{11.13}$$

where $\ddot{\mathbf{x}}^d$ is any vector of "desired" acceleration whose tangential component matches the tangential acceleration of the given trajectory, i.e. $\mathbf{P}_{/\!/}\ddot{\mathbf{x}}^d = \ddot{\mathbf{x}}_{/\!/}$.

Now we show how the decomposition (11.13) is related to and generalizes (11.2). This is based on the observation that (11.13) is the same as the solution to the least-squared problem below

$$\min_{\ddot{\mathbf{q}}} \frac{1}{2}\|\ddot{\mathbf{x}}^d - \ddot{\mathbf{x}}\|^2 \quad \text{s.t. } \ddot{\mathbf{x}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}. \tag{11.14}$$

This equivalence can be easily seen by resolving the constraint, setting the gradient of the resulting quadratic to zero, i.e.,

$$\mathbf{J}^\top\mathbf{J}\ddot{\mathbf{q}} + \mathbf{J}^\top\dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}^\top\ddot{\mathbf{x}}^d \tag{11.15}$$

and re-expressing the optimal solution in $\ddot{\mathbf{x}}$. This relationship demonstrates that a QP very similar in structure to (11.2). Both express dynamics of the forced system as $\left(\mathbf{J}^\top\mathbf{J}\right)\ddot{\mathbf{q}} + \mathbf{J}^\top\dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}^\top\mathbf{f}$ with $\mathbf{f} = \ddot{\mathbf{x}}^d$ (task space coordinates were chosen specifically so that weights (equiv. masses) would be 1, so forces and accelerations have the same units—different task coordinates would result in constant weights, corresponding to a notion of constant mass in the ambient space).

The Curvature Terms

As a side note, the above discussion offers insight into the term $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{J}^\top\dot{\mathbf{J}}\dot{\mathbf{q}}$. The term $\dot{\mathbf{J}}\dot{\mathbf{q}}$ captures components describing both curvature of the manifold through the ambient task space and components describing how the specific coordinate $\mathbf{q}$ (tangentially) curves across the sub-manifold $\mathcal{X}$. Explicitly, $\ddot{\mathbf{x}}^c = \dot{\mathbf{J}}\dot{\mathbf{q}}$ has units of acceleration in the ambient space and captures how the tangent space (given by the columns of $\mathbf{J}$) changes in the direction of motion. The acceleration $\ddot{\mathbf{x}}^c$ decomposes as $\ddot{\mathbf{x}}^c = \ddot{\mathbf{x}}^c_\perp + \ddot{\mathbf{x}}^c_{/\!/}$, into two orthogonal components consisting of a component perpendicular to the tangent space $\ddot{\mathbf{x}}^c_\perp = \mathbf{P}_\perp\ddot{\mathbf{x}}^c$ and

a component parallel to the tangent space $\ddot{\mathbf{x}}^c_{/\!/} = \mathbf{P}_{/\!/}\ddot{\mathbf{x}}^c$. The term $\mathbf{P}_\perp\dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{P}_\perp\ddot{\mathbf{x}}^c$ given in (11.12) extracts specifically the perpendicular component $\ddot{\mathbf{x}}^c_\perp$. The other component $\ddot{\mathbf{x}}^c_{/\!/}$ is, therefore, in a sense irrelevant to fundamental geometric behavior of the underlying system, and is only required when expressing the behavior in the specific coordinates $\mathbf{q}$. Indeed, when expressing the equations of motion in $\mathbf{q}$, the related term manifests as $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{J}^\top\dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}^\top\ddot{\mathbf{x}}^c_{/\!/}$ since $\mathbf{J}^\top\ddot{\mathbf{x}}^c_\perp = 0$, and depends only on the parallel component $\ddot{\mathbf{x}}^c_{/\!/}$. This observation emphasizes why we designate the term *fictitious forces*. Here and below we will consider these terms to be *curvature* terms as they compensate for curvature in the system coordinates.

11.3.4 Non-constant Weights and Implicit Task Spaces

Section 11.3.3 above derived the geometric perspective of equations of motion, but only for mechanical systems whose inertia matrix (equiv. Riemannian metric) can be expressed as $\mathbf{M}(\mathbf{q}) = \mathbf{J}(\mathbf{q})^\top\mathbf{J}(\mathbf{q})$ globally for some map $\mathbf{x} = \psi(\mathbf{q})$ with $\mathbf{J}(\mathbf{q}) = \partial_\mathbf{q}\phi(\mathbf{q})$. Fortunately, due to a deep and fundamental theorem proved by John Nash in 1956, called the Nash embedding theorem (Nash, 1956), *all* Riemannian manifolds, and hence *all* mechanical systems can be expressed this way, so the arguments of Section 11.3.3 hold without loss of generality for all mechanical systems. It is called an embedding theorem because the map $\mathbf{x} = \psi(\mathbf{q})$ acts to embed the manifold $\mathcal{C}$ into a higher-dimensional ambient Euclidean space where we can replace implicit geometry represented by the metric $\mathbf{M}(\mathbf{q})$ with an explicit sub-manifold in the ambient space.

This ambient representation is a convenient for understanding and visualizing the non-linear geometry of a mechanical system, but it is unfortunately often difficult, or even impossible, to find a closed form expression for a task map $\mathbf{x} = \psi(\mathbf{q})$ from a given metric $\mathbf{M}(\mathbf{q})$. We, therefore, cannot rely our ability to operate directly in the ambient space using the QP given in (11.14).

This subsection addresses that problem by deriving a QP expression analogous to (11.14),

but for which task space weights are general non-constant positive definite matrices (which we will see are the same as Riemannian metrics). We will arrive at this expression by considering again the ambient setting, but assuming that the unknown embedding can be decomposed in the composition of a known task space $\mathbf{x} = \psi(\mathbf{q})$ and then an known map from the task space to the ambient Euclidean space $\mathbf{z} = \zeta(\mathbf{x})$ described in the Nash's theorem. We suppose the priority weight is given as the induced Riemannian metric $\mathbf{G}(\mathbf{x}) = \mathbf{J}_\zeta(\mathbf{x})^\top \mathbf{J}_\zeta(\mathbf{x})$ on $\mathbf{x}$ defined by the second map ζ. We note that the final result will be expressed entirely in terms of $\mathbf{G}(\mathbf{x})$, so it can be used without explicit knowledge of ζ.

Suppose we have a Riemannian metric (equiv. inertia matrix) $\mathbf{M}$ which decomposes as $\mathbf{M} = \mathbf{J}^\top \mathbf{J}$, where $\mathbf{J} = \mathbf{J}_\zeta \mathbf{J}_\phi$ is the Jacobian of the composite map $\mathbf{z} = \zeta \circ \phi(\mathbf{q})$ which itself consists of two parts $\mathbf{x} = \phi(\mathbf{q})$ and $\mathbf{z} = \zeta(\mathbf{x})$. Because the intermediate task space metric is $\mathbf{G} = \mathbf{J}_\zeta^\top \mathbf{J}_\zeta$, denote the task space force as $\mathbf{f}_\mathbf{x} = \mathbf{J}_\zeta^\top \mathbf{f}_\mathbf{z}$ and by (11.15) we have

$$
\begin{aligned}
&\mathbf{J}^\top \mathbf{J}\ddot{\mathbf{q}} + \mathbf{J}^\top \dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}^\top \mathbf{f}_\mathbf{z} \\
\Rightarrow\ & \left(\mathbf{J}_\phi^\top \left(\mathbf{J}_\zeta^\top \mathbf{J}_\zeta\right) \mathbf{J}_\phi\right) \ddot{\mathbf{q}} + \mathbf{J}_\phi^\top \mathbf{J}_\zeta^\top \frac{d}{dt} \left(\mathbf{J}_\zeta \mathbf{J}_\phi\right) \dot{\mathbf{q}} = \mathbf{J}_\phi^\top \mathbf{J}_\zeta^\top \mathbf{f}_\mathbf{z} \\
\Rightarrow\ & \mathbf{J}_\phi^\top \mathbf{G} \mathbf{J}_\phi \ddot{\mathbf{q}} + \mathbf{J}_\phi^\top \mathbf{J}_\zeta^\top \left(\dot{\mathbf{J}}_\zeta \mathbf{J}_\phi + \mathbf{J}_\zeta \dot{\mathbf{J}}_\phi\right) \dot{\mathbf{q}} = \mathbf{J}_\phi^\top \mathbf{f}_\mathbf{x} \\
\Rightarrow\ & \mathbf{J}_\phi^\top \mathbf{G} \mathbf{J}_\phi \ddot{\mathbf{q}} + \mathbf{J}_\phi^\top \left(\mathbf{J}_\zeta^\top \dot{\mathbf{J}}_\zeta \dot{\mathbf{x}}\right) + \mathbf{J}_\phi^\top \left(\mathbf{J}_\zeta^\top \mathbf{J}_\zeta\right) \dot{\mathbf{J}}_\phi \dot{\mathbf{q}} = \mathbf{J}_\phi^\top \mathbf{f}_\mathbf{x} \\
\Rightarrow\ & \mathbf{J}_\phi^\top \mathbf{G} \mathbf{J}_\phi \ddot{\mathbf{q}} + \mathbf{J}_\phi^\top \mathbf{G} \dot{\mathbf{J}}_\phi \dot{\mathbf{q}} = \mathbf{J}_\phi^\top \left(\mathbf{f}_x - \boldsymbol{\xi}_\mathbf{G}\right).
\end{aligned}
$$

where we recall $\boldsymbol{\xi}_\mathbf{G}$ in given in (11.6). Rearranging that final expression and denoting $\underline{\ddot{\mathbf{x}}}^d = \ddot{\mathbf{x}}^d - \mathbf{G}^{-1}\boldsymbol{\xi}_\mathbf{G}$ with $\ddot{\mathbf{x}}^d = \mathbf{G}^{-1}\mathbf{f}_\mathbf{x}$, we can write the above equation as

$$
\mathbf{J}_\phi^\top \mathbf{G} \left(\underline{\ddot{\mathbf{x}}}^d - \left(\mathbf{J}_\phi \ddot{\mathbf{q}} + \dot{\mathbf{J}}_\phi \dot{\mathbf{q}}\right)\right) = 0, \tag{11.16}
$$

which is the first-order optimality condition of the QP

$$\min_{\ddot{\mathbf{q}}} \frac{1}{2}\|\underline{\ddot{\mathbf{x}}}^d - \ddot{\mathbf{x}}\|^2_{\mathbf{G}} \qquad \text{s.t. } \ddot{\mathbf{x}} = \mathbf{J}_\phi \ddot{\mathbf{q}} + \dot{\mathbf{J}}_\phi \dot{\mathbf{q}}. \tag{11.17}$$

This QP is expressed in terms of the task map $\mathbf{x} = \psi(\mathbf{q})$, the task space metric $\mathbf{G}(\mathbf{x})$, the task space desired accelerations $\ddot{\mathbf{x}}^d$, and the curvature term $\boldsymbol{\xi}_{\mathbf{G}}$ derived from the task space metric $\mathbf{G}(\mathbf{x})$. The QP follows a very similar pattern to the QPs described above, but this time the priority weight matrix $\mathbf{G}$ is a non-constant function of $\mathbf{x}$. The one modification required to reach this matching form is to augment the desired acceleration $\ddot{\mathbf{x}}^d$ with the curvature term $\boldsymbol{\xi}_{\mathbf{G}}$ calculated from $\mathbf{G}$ using (11.6) to get the target $\underline{\ddot{\mathbf{x}}}^d = \ddot{\mathbf{x}}^d - \mathbf{G}^{-1}\boldsymbol{\xi}_{\mathbf{G}}$. Importantly, while we start by assuming the map $\mathbf{z} = \zeta(\mathbf{x})$, at the end we show that we actually only need to know $\mathbf{G}$.

11.3.5 Limitations of geometric control

Even with the tools of geometric mechanics, the final QP given in (11.17) can still only express task priority weights as positive definite matrices that vary as a function of configuration (i.e. position). Frequently, more nuanced control over those priorities is crucial. For instance, collision avoidance tasks should activate when the control-point is close to an obstacle and heading toward it, but they should deactivate either when the control-point is far from the obstacle *or* when it's moving away from the obstacle, regardless of its proximity. Importantly, reducing the desired acceleration to zero in these cases is not enough—when these tasks deactivate, they should drop entirely from the equation rather than voting with high weight for zero acceleration. Enabling priorities vary as a function of the full robot state (configuration *and* velocity) is therefore paramount.

The theory of RMPflow and GDSs developed below generalizes geometric mechanics to enable expressing these more nuanced priority matrices while maintaining stability. Additionally, since geometric control theory itself is quite abstract, we build on results re-

ducing the calculations to recursive least squares similar to that given in Section 11.3.4 to derive a concrete tree data structure to aid in the design of controllers within this energy shaping framework.

11.4 RMPflow

RMPflow is an efficient manifold-oriented computational graph for automatic generation of motion policies that can tackle multiple task specifications. Let $\mathcal{T}_{l_i}$ denote the ith subtask. More precisely, RMPflow is aimed for problems with a task space $\mathcal{T} = \{\mathcal{T}_{l_i}\}$ that is related to the configuration space $\mathcal{C}$ through a tree-structured task map ψ, in which $\mathcal{C}$ is the root node and the subtask spaces $\{\mathcal{T}_{l_i}\}$ are the leaf nodes. Given user-specified motion policies $\{\pi_{l_i}\}$ on the subtask spaces $\{\mathcal{T}_{l_i}\}$ as RMPs, RMPflow is designed to *consistently* combine these subtask policies into a global policy π on $\mathcal{C}$.

To this end, RMPflow introduces 1) a data structure, called the *RMP-tree*, to describe the tree-structured task map ψ and the policies, and 2) a set of operators, called the *RMP-algebra*, to propagate information across the RMP-tree. To compute $\pi(\mathbf{q}(t), \dot{\mathbf{q}}(t))$ at time t, RMPflow operates in two steps: it first performs a *forward pass* to propagate the state from the root node (i.e., $\mathcal{C}$) to the leaf nodes (i.e., $\{\mathcal{T}_{l_i}\}$); then it performs a *backward pass* to propagate the RMPs from the leaf nodes to the root node while tracking their geometric information to achieve consistency. These two steps are realized by recursive use of RMP-algebra, exploiting shared computation paths arising from the tree structure to maximize efficiency.

In the following, we describe the details of RMPflow and give some useful examples of subtask motion policies.

11.4.1 Structured Task Maps

In many applications, the task-space manifold $\mathcal{T}$ is structured. In this chapter, we consider the case where the task map ψ can be expressed through a tree-structured composition

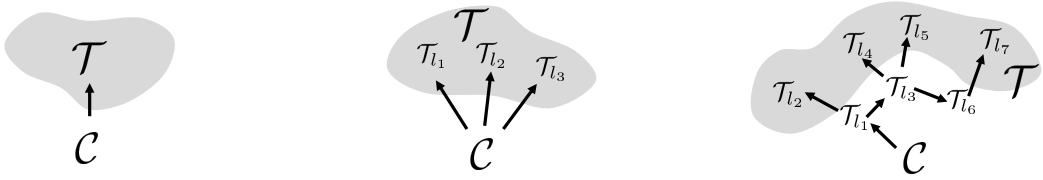


Figure 11.1: Tree-structured task maps

of transformations $\{\psi_{e_i}\}$, where ψ_{e_i} is the ith transformation. Fig. 11.1 illustrates some common examples, where each node denotes a manifold and each edge denotes a transformation. This family trivially includes the unstructured task space $\mathcal{T}$ (Fig. 11.1a) and the product manifold $\mathcal{T} = \mathcal{T}_{l_1} \times \cdots \times \mathcal{T}_{l_K}$ (Fig. 11.1b), where K is the number of subtasks. A more interesting example is the kinematic tree (Fig. 11.1c), where the task map considers the relationship between the configuration space $\mathcal{C}$ (the root node) and a collection of subtask spaces (the leaf nodes) that describe, e.g., the tracking and obstacle avoidance tasks along a multi-DOF robot.

The main motivation of explicitly handling the structure in the task map ψ is two-fold. First, it allows RMPflow to exploit computation shared across different subtask maps. Second, it allows the user to focus on designing motion policies for each subtask individually, which is easier than directly designing a global policy for the entire task space $\mathcal{T}$. For example, $\mathcal{T}$ may describe the problem of humanoid walking, which includes staying balanced, scheduling contacts, and avoiding collisions. Directly parameterizing a policy to satisfy all these objectives can be daunting, whereas designing a policy for each subtask is more feasible.

11.4.2 Riemannian Motion Policies (RMPs)

Knowing the structure of the task map is not sufficient for consistently combining subtask policies: we require some information about the motion policies' behaviors (Ratliff, Issac, and Kappler, 2018). Toward this end, we adopt an abstract description of motion policies, called RMPs (Ratliff, Issac, and Kappler, 2018), for the nodes of the RMP-tree. An RMP describes a second-order differential equation along with its geometric information on a

smooth manifold.

Specifically, let $\mathcal{M}$ be an m-dimensional manifold with coordinate $\mathbf{x} \in \mathbb{R}^m$. The *canonical form* of an RMP on $\mathcal{M}$ is a pair $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$, where $\mathbf{a} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is a continuous motion policy and $\mathbf{M} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{m \times m}_+$ is a differentiable map. Borrowing terminology from mechanics, we call $\mathbf{a}(\mathbf{x}, \dot{\mathbf{x}})$ the *desired acceleration* and $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ the *inertia matrix* at $(\mathbf{x}, \dot{\mathbf{x}})$, respectively.[6] For now, we can intuitively think that $\mathbf{M}$ defines the directional importance of $\mathbf{a}$ when it is combined with other motion policies. Later in Section 11.5, we will show that $\mathbf{M}$ is closely related to the concept of Riemannian metric, which describes how the space is stretched along the curve generated by $\mathbf{a}$; when $\mathbf{M}$ depends on the state, the space becomes *non-Euclidean*.

In this chapter, we additionally introduce a new RMP form, called the *natural form*. Given an RMP in its canonical form $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$, we define another pair as its natural form $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$, where $\mathbf{f} = \mathbf{M}\mathbf{a}$ is the *desired force* map. While the transformation between these two forms may look trivial, their distinction will be useful later when we introduce the RMP-algebra.

11.4.3 RMP-tree

The RMP-tree is the core data structure used by RMPflow. An RMP-tree is a directed tree, in which each node represents an RMP and its state, and each edge corresponds to a transformation between manifolds. The root node of the RMP-tree describes the global policy π on $\mathcal{C}$, and the leaf nodes describe the local policies $\{\pi_{l_i}\}$ on $\{\mathcal{T}_{l_i}\}$.

To illustrate, let us consider a node u and its K child nodes $\{v_i\}_{i=1}^K$. Suppose u describes an RMP $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ and v_i describes an RMP $[\mathbf{f}_i, \mathbf{M}_i]^{\mathcal{N}_i}$, where $\mathcal{N}_i = \psi_{e_i}(\mathcal{M})$ for some ψ_{e_i}. Then we write $u = ((\mathbf{x}, \dot{\mathbf{x}}), [\mathbf{f}, \mathbf{M}]^{\mathcal{M}})$ and $v_i = ((\mathbf{y}_i, \dot{\mathbf{y}}_i), [\mathbf{f}_i, \mathbf{M}_i]^{\mathcal{N}_i})$, where $(\mathbf{x}, \dot{\mathbf{x}})$ is the state of u, and $(\mathbf{y}_i, \dot{\mathbf{y}}_i)$ is the state of v_i; the edge connecting u and v_i points from u to v_i along ψ_{e_i}. We will continue to use this example to illustrate how RMP-algebra propagates

[6] Here we adopt a slightly different terminology from (Ratliff, Issac, and Kappler, 2018). We note that $\mathbf{M}$ and $\mathbf{f}$ do not necessarily correspond to the inertia and force of a physical mechanical system.

the information across the RMP-tree.

11.4.4 RMP-algebra

The RMP-algebra of RMPflow consists of three operators (`pushforward`, `pullback`, and `resolve`) to propagate information across the RMP-tree.[7] They form the basis of the forward and backward passes for automatic policy generation, described in the next section.

1. `pushforward` is the operator to forward propagate the *state* from a parent node to its child nodes. Using the previous example, given $(\mathbf{x}, \dot{\mathbf{x}})$ from u, it computes $(\mathbf{y}_i, \dot{\mathbf{y}}_i) = (\psi_{e_i}(\mathbf{x}), \mathbf{J}_i(\mathbf{x})\dot{\mathbf{x}})$ for each child node v_i, where $\mathbf{J}_i = \partial_{\mathbf{x}}\psi_{e_i}$ is a Jacobian matrix. The name "pushforward" comes from the linear transformation of tangent vector $\dot{\mathbf{x}}$ to the image tangent vector $\dot{\mathbf{y}}_i$.

2. `pullback` is the operator to backward propagate the natural-formed RMPs from the child nodes to the parent node. It is done by setting $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ with

$$\mathbf{f} = \sum_{i=1}^{K} \mathbf{J}_i^{\top}(\mathbf{f}_i - \mathbf{M}_i\dot{\mathbf{J}}_i\dot{\mathbf{x}}), \mathbf{M} = \sum_{i=1}^{K} \mathbf{J}_i^{\top}\mathbf{M}_i\mathbf{J}_i \tag{11.18}$$

   The name "pullback" comes from the linear transformations of the cotangent vector (1-form) $\mathbf{f}_i - \mathbf{M}_i\dot{\mathbf{J}}_i\dot{\mathbf{x}}$ and the inertia matrix (2-form) $\mathbf{M}_i$. In summary, velocities can be pushfowarded along the direction of ψ_i, and forces and inertial matrices can be pullbacked in the opposite direction.

   To gain more intuition of `pullback`, we write `pullback` in the canonical form of RMPs. It can be shown that the canonical form $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ of the natural form $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ above is the solution to a least-squares problem:

$$\mathbf{a} = \arg\min_{\mathbf{a}'} \frac{1}{2} \sum_{i=1}^{K} \|\mathbf{J}_i\mathbf{a}' + \dot{\mathbf{J}}_i\dot{\mathbf{x}} - \mathbf{a}_i\|^2_{\mathbf{M}_i}$$

[7] Precisely they propagate the numerical values of RMPs and states at a particular time.

$$= (\sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i)^\dagger (\sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i (\mathbf{a}_i - \dot{\mathbf{J}}_i \dot{\mathbf{x}})) \tag{11.19}$$

where $\mathbf{a}_i = \mathbf{M}_i^\dagger \mathbf{f}_i$ and $\|\cdot\|_{\mathbf{M}_i}^2 = \langle \cdot, \mathbf{M}_i \cdot \rangle \cdot^\top \mathbf{M}_i \cdot$. Because $\ddot{\mathbf{y}}_i = \mathbf{J}_i \ddot{\mathbf{x}} + \dot{\mathbf{J}}_i \dot{\mathbf{x}}$, `pullback` attempts to find an $\mathbf{a}$ that can realize the desired accelerations $\{\mathbf{a}_i\}$ while trading off approximation errors with an importance weight defined by the inertia matrix $\mathbf{M}_i(\mathbf{y}_i, \dot{\mathbf{y}}_i)$. The use of state dependent importance weights is a distinctive feature of RMPflow. It allows RMPflow to activate different RMPs according to *both* configuration and velocity (see Section 11.4.6 for examples). Finally, we note that the `pullback` operator defined in this chapter is slightly different from the original definition given in (Ratliff, Issac, and Kappler, 2018), which ignores the term $\dot{\mathbf{J}}_i \dot{\mathbf{x}}$ in (11.19). While ignoring $\dot{\mathbf{J}}_i \dot{\mathbf{x}}$ does not necessary destabilize the system (Lo, Cheng, and Huang, 2016), its inclusion is critical to implement consistent policy behaviors.

3. `resolve` is the last operator of RMP-algebra. It maps an RMP from its natural form to its canonical form. Given $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$, it outputs $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ with $\mathbf{a} = \mathbf{M}^\dagger \mathbf{f}$, where $\dagger$ denotes Moore-Penrose inverse. The use of pseudo-inverse is because in general the inertia matrix is only positive semi-definite. Therefore, we also call the natural form of $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ the *unresolved form*, as potentially it can be realized by multiple RMPs in the canonical form.

11.4.5 Algorithm: Motion Policy Generation

Now we show how RMPflow uses the RMP-tree and RMP-algebra to generate a global policy π on $\mathcal{C}$. Suppose each subtask policy is provided as an RMP in the natural form. First, we construct an RMP-tree with the same structure as ψ, where we assign subtask RMPs as the leaf nodes and the global RMP $[\mathbf{f}_r, \mathbf{M}_r]^{\mathcal{C}}$ as the root node. With the RMP-tree specified, RMPflow can perform automatic policy generation. At every time instance, it first performs a forward pass: it recursively calls `pushforward` from the root node to the

leaf nodes to update the state information in each node in the RMP-tree. Second, it performs a backward pass: it recursively calls `pullback` from the leaf nodes to the root node to back propagate the values of the RMPs in the natural form, and finally calls `resolve` at the root node to transform the global RMP $[\mathbf{f}_r, \mathbf{M}_r]^{\mathcal{C}}$ into its canonical form $(\mathbf{a}_r, \mathbf{M}_r)^{\mathcal{C}}$ for policy execution (i.e. setting $\pi(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{a}_r$).

The process of policy generation of RMPflow uses the tree structure for computational efficiency. For K subtasks, it has time complexity $O(K)$ in the worst case[8] as opposed to $O(K \log K)$ of a naive implementation which does not exploit the tree structure. Furthermore, all computations of RMPflow are carried out using matrix-multiplications, except for the final `resolve` call, because the RMPs are expressed in the natural form in `pullback` instead of the canonical form suggested originally in (Ratliff, Issac, and Kappler, 2018). This design makes RMPflow numerically stable, as only one matrix inversion $\mathbf{M}_r^{\dagger}\mathbf{f}_r$ is performed at the root node with both $\mathbf{f}_r$ and $\mathbf{M}_r$ in the span of the same Jacobian matrix due to `pullback`.

11.4.6 Example RMPs

We give a quick overview of some RMPs useful in practice (see Appendix D of the technical report (Cheng et al., 2018c) for further discussion of these RMPs) We recall from (11.19) that $\mathbf{M}$ dictates the directional importance of an RMP.

Collision/joint limit avoidance

Barrier-type RMPs are examples that use velocity dependent inertia matrices, which can express importance as a function of robot heading (a property that traditional mechanical principles fail to capture). Here we demonstrate a collision avoidance policy in the 1D distance space $x = d(\mathbf{q})$ to an obstacle. Let $g(x, \dot{x}) = w(x)u(\dot{x}) > 0$ for some functions w and u. We consider a motion policy such that $m(x, \dot{x})\ddot{x} + \frac{1}{2}\dot{x}^2\partial_x g(x, \dot{x}) = -\partial_x \Phi(x) - b\dot{x}$

[8] The case with a binary tree.

and define its inertia matrix $m(x, \dot{x}) = g(x, \dot{x}) + \frac{1}{2}\dot{x}\partial_{\dot{x}} g(x, \dot{x})$, where Φ is a potential and $b > 0$ is a damper. We choose $w(x)$ to increase as x decreases (close to the obstacle), $u(\dot{x})$ to increase when $\dot{x} < 0$ (moving toward the obstacle), and $u(\dot{x})$ to be constant when $\dot{x} \geq 0$. With this choice, the RMP can be turned off in `pullback` when the robot heads away from the obstacle. This motion policy is a GDS and g is its metric (cf. Section 12.2.2); the terms $\frac{1}{2}\dot{x}\partial_{\dot{x}} g(x, \dot{x})$ and $\frac{1}{2}\dot{x}^2\partial_x g(x, \dot{x})$ are due to non-Euclidean geometry and produce natural repulsive behaviors as the robot moves toward the obstacle, and little or no force when it starts to move away.

Target attractors

Designing an attractor policy is relatively straightforward. For a task space with coordinate $\mathbf{x}$, we can consider an inertia matrix $\mathbf{M}(\mathbf{x}) \succ 0$ and a motion policy such that $\ddot{\mathbf{x}} = -\nabla\widetilde{\Phi} - \beta(\mathbf{x})\dot{\mathbf{x}} - \mathbf{M}^{-1}\boldsymbol{\xi}_{\mathbf{M}}$, where $\widetilde{\Phi}(\mathbf{x}) \approx \|\mathbf{x}\|$ is a smooth attractor potential, $\beta(\mathbf{x}) \geq 0$ is a damper, and $\boldsymbol{\xi}_{\mathbf{M}}$ is a curvature term due to $\mathbf{M}$. It can be shown that this differential equation is also a GDS Cheng et al., 2018c, Appendix D.

Orientations

As RMPflow directly works with manifold objects, orientation controllers become straightforward to design, independent of the choice of coordinate (cf. Section 11.5.4). For example, we can define RMPs on a robotic link's surface in any preferred coordinate (e.g. in one or two axes attached to an arbitrary point) with the above described attractor to control the orientation. This follows a similar idea outlined in the Appendix of (Ratliff, Issac, and Kappler, 2018).

Q-functions

Perhaps surprising, RMPs can be constructed using Q-functions as metrics (we invite readers to read (Ratliff, Issac, and Kappler, 2018) for details on how motion optimizers can be

reduced to Q-functions and the corresponding RMPs). While these RMPs may not satisfy the conditions of a GDS that we later analyze, they represent a broader class of RMPs that leads to substantial benefits (e.g. escaping local minima) in practice. Also, Q-functions are closely related to Lyapunov functions and geometric control (Lewis, 2000); we will further explore this direction in future work.

11.5 Theoretical Analysis of RMPflow

We investigate the properties of RMPflow when the child-node motion policies belong to a class of differential equations, which we call *structured geometric dynamical systems* (structured GDSs). We present the following results.

1. **Closure**: We show that the `pullback` operator retains a closure of structured GDSs. When the child-node motion policies are structured GDSs, the parent-node dynamics also belong to the same class.

2. **Stability**: Using the closure property, we provide sufficient conditions for the feedback policy of RMPflow to be stable. In particular, we cover a class of dynamics with *velocity-dependent* metrics that are new to the literature.

3. **Invariance**: As its name suggests, RMPflow is closely related to differential geometry. We show that RMPflow is intrinsically coordinate-free. This means that a set of subtask RMPs designed for one robot can be transferred to another robot while maintaining the same task-space behaviors.

Setup Below we consider the manifolds in the nodes of the RMP-tree to be finite-dimensional and smooth. Without loss of generality, for now we assume that each manifold can be described in a single chart (i.e. using a global coordinate), so that we can write down the equations concretely using finite-dimensional variables. This restriction will be removed when we presents the coordinate-free form in Section 11.5.4. We also assume that all the

maps are sufficiently smooth so the required derivatives are well defined. The proofs of this section can found in Section 11.B.

11.5.1 Geometric Dynamical Systems (GDSs)

We first define a new family of dynamics, called GDSs, useful to specify RMPs on manifolds. (Structured GDSs will be introduced shortly in the next section.) At a high-level, a GDS can be thought as a virtual mechanical system defined on a manifold with an inertia that depends on *both* configuration and velocity. Formally, let us consider an m-dimensional manifold $\mathcal{M}$ with chart $(\mathcal{M}, \mathbf{x})$ (i.e. a coordinate system on $\mathcal{M}$). Let $\mathbf{G} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m \times m}$, $\mathbf{B} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m \times m}$, and $\Phi : \mathbb{R}^m \to \mathbb{R}$ be sufficiently smooth functions. We say a dynamical system on $\mathcal{M}$ is a *GDS* $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$, if it satisfies the differential equation

$$(\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}))\, \ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}, \tag{11.20}$$

where we define

$$\begin{aligned}
\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) &\coloneqq \frac{1}{2} \sum_{i=1}^{m} \dot{x}_i \partial_{\dot{\mathbf{x}}} \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}}) \\
\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) &\coloneqq \overset{\mathbf{x}}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} - \frac{1}{2} \nabla_{\mathbf{x}} (\dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}) \\
\overset{\mathbf{x}}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) &\coloneqq [\partial_{\mathbf{x}} \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}]_{i=1}^{m}
\end{aligned}$$

and $\mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})$ is the ith column of $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$. We refer to $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ as the *metric* matrix, $\mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})$ as the *damping* matrix, and $\Phi(\mathbf{x})$ as the *potential* function which is lower-bounded. In addition, we call the term in front of $\ddot{\mathbf{x}}$ in (13.3),

$$\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}), \tag{11.21}$$

the *inertia* matrix of GDS $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$, which can be asymmetric. When $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ is nonsingular, we say the GDS is *non-degenerate*. We will assume (13.3) is non-degenerate for now so that it uniquely defines a differential equation. The discussion on the general case is postponed to Section 11.C.

In GDSs, $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ induces a metric of $\dot{\mathbf{x}}$, measuring its length as $\frac{1}{2}\dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$. When $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ depends on $\mathbf{x}$ and $\dot{\mathbf{x}}$, it also induces non-trivial *curvature* terms $\boldsymbol{\Xi}(\mathbf{x}, \dot{\mathbf{x}})$ and $\boldsymbol{\xi}(\mathbf{x}, \dot{\mathbf{x}})$. In a particular case when $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$ (i.e. it depends on configuration only), the GDSs reduce to the widely studied *simple mechanical systems* (SMSs) in geometric mechanics (Bullo and Lewis, 2004) (see also (11.5))

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \nabla_{\mathbf{x}}\Phi(\mathbf{x}) = -\mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} \tag{11.22}$$

where the Coriolis force $\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$ can be shown equal to $\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})$. In this special case, we have $\mathbf{M}(\mathbf{x}) = \mathbf{G}(\mathbf{x})$, i.e., the inertia matrix is the same as the metric matrix (this is exactly the finding in geometric mechanics discussed in Section 11.3.3) We will revisit the connection between GDSs and SMSs again in Section 11.6.1 (and show why $\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} = \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})$) after the analysis of geometric properties of GDSs. For now, we can think of GDSs as generalization of SMSs to have inertia $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ and metric $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ that also change with velocity! This velocity-dependent extension is important and non-trivial. As discussed in earlier Section 11.4.6, it generalizes the dynamics of classical rigid-body systems, allowing the space to morph according to the velocity direction.

Finally, as its name hints, GDSs possess geometric properties. Particularly, when $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ is invertible, the left-hand side of (13.3) is related to a quantity $\mathbf{a}_{\mathbf{G}} = \ddot{\mathbf{x}} + \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})^{-1}(\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}))$, known as the *geometric acceleration* (cf. Section 11.5.4). (Therefore these terms must not be separated; e.g. $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}}$ alone may not possess particular meaning.) In other words, we can think of (13.3) as setting $\mathbf{a}_{\mathbf{G}}$ along the negative natural gradient $-\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})^{-1}\nabla_{\mathbf{x}}\Phi(\mathbf{x})$ while imposing damping $-\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})^{-1}\mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$.

11.5.2 Closure

Earlier, we argued vaguely that by tracking the geometry in `pullback` in (11.18) through propagating RMPs instead of just motion policies, the task properties can be preserved. Here, we formalize this consistency concept of RMPflow as a closure of differential equations, named structured GDSs. Structured GDSs augment GDSs with information on how the metric matrix $\mathbf{G}$ factorizes. We call such information a *structure*. Specifically, suppose $\mathbf{G}$ has a structure $\mathcal{S}$ that factorizes $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{J}(\mathbf{x})^\top \mathbf{H}(\mathbf{y}, \dot{\mathbf{y}}) \mathbf{J}(\mathbf{x})$, where $\mathbf{y} : \mathbf{x} \mapsto \mathbf{y}(\mathbf{x}) \in \mathbb{R}^n$ and $\mathbf{H} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+^{n \times n}$, and $\mathbf{J}(\mathbf{x}) = \partial_{\mathbf{x}} \mathbf{y}$ is the Jacobian. We say a dynamical system on $\mathcal{M}$ is a *structured GDS* $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ if it satisfies the differential equation

$$\left(\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\right) \ddot{\mathbf{x}} + \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}}) \dot{\mathbf{x}} \tag{11.23}$$

where $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \mathbf{J}(\mathbf{x})^\top (\boldsymbol{\xi}_{\mathbf{H}}(\mathbf{y}, \dot{\mathbf{y}}) + (\mathbf{H}(\mathbf{y}, \dot{\mathbf{y}}) + \boldsymbol{\Xi}_{\mathbf{H}}(\mathbf{y}, \dot{\mathbf{y}})) \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}}) \dot{\mathbf{x}})$. If we compare GDSs in (13.3) and structured GDSs in (12.7), the difference is that $\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})$ is now replaced by a different curvature term $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x}, \dot{\mathbf{x}})$ that is defined by *both* the metric and factorization. In fact, GDSs are structured GDSs with a *trivial* structure (i.e. $\mathbf{y} = \mathbf{x}$). Also, one can easily show that structured GDSs reduce to GDSs (i.e. the structure offers no extra information) if $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$, or if $n, m = 1$. Given two structures, we say $\mathcal{S}_a$ *preserves* $\mathcal{S}_b$ if $\mathcal{S}_a$ has the factorization (of $\mathbf{H}$) made by $\mathcal{S}_b$. In Section 11.5.4, we will show that structured GDSs are related to a geometric object, pullback connection, which turns out to be the coordinate-free version of `pullback`.

Below we show the closure property: when the children of a parent node are structured GDSs, the parent node defined by `pullback` is also a structured GDS with respect to the pullbacked structured metric matrix, damping matrix, and potentials. Without loss of generality, we consider again a parent node on $\mathcal{M}$ with K child nodes on $\{\mathcal{N}_i\}_{i=1}^K$. We note that $\mathbf{G}_i$ and $\mathbf{B}_i$ can be functions of both $\mathbf{y}_i$ and $\dot{\mathbf{y}}_i$.

Theorem 11.5.1. *Let the ith child node follow $(\mathcal{N}_i, \mathbf{G}_i, \mathbf{B}_i, \Phi_i)_{\mathcal{S}_i}$ and have coordinate $\mathbf{y}_i$.*

Let $\mathbf{f}_i = -\boldsymbol{\eta}_{\mathbf{G}_i;\mathcal{S}_i} - \nabla_{\mathbf{y}_i}\Phi_i - \mathbf{B}_i\dot{\mathbf{y}}_i$ *and* $\mathbf{M}_i = \mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i}$*. If* $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ *of the parent node is given by* `pullback` *with* $\{[\mathbf{f}_i, \mathbf{M}_i]^{\mathcal{N}_i}\}_{i=1}^K$ *and* $\mathbf{M}$ *is non-singular, the parent node follows the* pullback structured GDS $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$*, where* $\mathbf{G} = \sum_{i=1}^K \mathbf{J}_i^\top \mathbf{G}_i \mathbf{J}_i$*,* $\mathbf{B} = \sum_{i=1}^K \mathbf{J}_i^\top \mathbf{B}_i \mathbf{J}_i$*,* $\Phi = \sum_{i=1}^K \Phi_i \circ \mathbf{y}_i$*,* $\mathcal{S}$ *preserves* $\mathcal{S}_i$*, and* $\mathbf{J}_i = \partial_{\mathbf{x}}\mathbf{y}_i$*. In other words, the parent node is the RMP* $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ *where* $\mathbf{M} = \sum_{i=1}^K \mathbf{J}_i^\top (\mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i})\mathbf{J}_i$ *and*

$$\mathbf{a} = (\mathbf{G} + \boldsymbol{\Xi}_{\mathbf{G}})^\dagger \left(-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} - \nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}}\right)$$

Particularly, if every $\mathbf{G}_i$ *is velocity-free and the child nodes are GDSs, the parent node follows* $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$*.*

Theorem 11.5.1 shows structured GDSs are closed under `pullback`. It means that the differential equation of a structured GDS with a tree-structured task map can be computed by recursively applying `pullback` from the leaves to the root, because in each recursive step, the form of structured GDS is preserved by `pullback`. Particularly, when $\mathbf{G}$ is velocity-free, one can show that `pullback` also preserves GDSs. We summarize these properties below.

Corollary 11.5.1. *If all leaf nodes follow GDSs and* $\mathbf{M}_r$ *at the root node is nonsingular, then the root node follows* $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ *as recursively defined by Theorem 11.5.1.*

11.5.3 Stability

By the closure property above, we analyze the stability of RMPflow when the leaf nodes are (structured) GDSs. For compactness, we will abuse the notation to write $\mathbf{M} = \mathbf{M}_r$. Suppose $\mathbf{M}$ is nonsingular and let $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ be the resultant structured GDS at the root node. We consider a Lyapunov function candidate

$$V(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \Phi(\mathbf{q}) \tag{11.24}$$

and derive its rate using properties of structured GDSs.

Proposition 11.5.1. *For $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$, it holds that $\dot{\mathbf{V}}(\mathbf{q}, \dot{\mathbf{q}}) = -\dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$.*

Proposition 11.5.1 directly implies the stability of structured GDSs by invoking LaSalle's invariance principle (Khalil, 1996). Here we summarize the result without proof.

Corollary 11.5.2. *For $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$, if $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}), \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$, the system converges to a forward invariant set $\mathcal{C}_\infty \coloneqq \{(\mathbf{q}, \dot{\mathbf{q}}) : \nabla_{\mathbf{q}}\Phi(\mathbf{q}) = 0, \dot{\mathbf{q}} = 0\}$.*

To show the stability of RMPflow, we need to further check when the assumptions in Corollary 11.5.2 hold. The condition $\mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$ is easy to satisfy: by Theorem 11.5.1, $\mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})$ has the form $\sum_{i=1}^{K} \mathbf{J}_i(\mathbf{q})^\top \mathbf{B}_i(\mathbf{x}_i, \dot{\mathbf{x}}_i)\mathbf{J}_i(\mathbf{q})$. Therefore, it automatically satisfies $\mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succeq 0$; to strictly ensure definiteness, we can copy $\mathcal{C}$ into an additional child node with a (small) positive-definite damping matrix. The condition on $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$ can be satisfied based on a similar argument about $\mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})$. In addition, we need to verify the assumption that $\mathbf{M}$ is nonsingular. Here we provide a sufficient condition. When satisfied, it implies the global stability of RMPflow in the sense of Corollary 11.5.2.

Theorem 11.5.2. *Suppose every leaf node is a GDS with a metric matrix in the form $\mathbf{R}(\mathbf{x}) + \mathbf{L}(\mathbf{x})^\top \mathbf{D}(\mathbf{x}, \dot{\mathbf{x}})\mathbf{L}(\mathbf{x})$ for differentiable functions $\mathbf{R}$, $\mathbf{L}$, and $\mathbf{D}$ satisfying $\mathbf{R}(\mathbf{x}) \succeq 0$, $\mathbf{D}(\mathbf{x}, \dot{\mathbf{x}}) = \operatorname{diag}((d_i(\mathbf{x}, \dot{y}_i))_{i=1}^n) \succeq 0$, and $\dot{y}_i \partial_{\dot{y}_i} d_i(\mathbf{x}, \dot{y}_i) \geq 0$, where $\mathbf{x}$ is the coordinate of the leaf-node manifold and $\dot{\mathbf{y}} = \mathbf{L}\dot{\mathbf{x}} \in \mathbb{R}^n$. It holds $\mathbf{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) \succeq 0$. If further $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}), \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$, then $\mathbf{M} \in \mathbb{R}^{d \times d}_{++}$, and the global RMP generated by RMPflow converges to the forward invariant set $\mathcal{C}_\infty$ in Corollary 11.5.2.*

A particular condition in Theorem 11.5.2 is when all the leaf nodes with velocity dependent metric are 1D. Suppose $x \in \mathbb{R}$ is its coordinate and $g(x, \dot{x})$ is its metric matrix. The sufficient condition essentially boils down to $g(x, \dot{x}) \geq 0$ and $\dot{x}\partial_{\dot{x}} g(x, \dot{x}) \geq 0$. This means that, given any $x \in \mathbb{R}$, $g(x, 0) = 0$, $g(x, \dot{x})$ is non-decreasing when $\dot{x} > 0$, and non-increasing when $\dot{x} < 0$. This condition is satisfied by the collision avoidance policy in Section 11.4.6.

11.5.4 Invariance

We now discuss the coordinate-free geometric properties of $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ generated by RMPflow. Due to space constraint, we only summarize the results. Here we assume that $\mathbf{G}$ is positive-definite.

We first introduce some additional notations for the coordinate-free analysis and give definitions of common differential geometric objects (please see, e.g., (Lee, 2009) for an excellent tutorial). For a manifold $\mathcal{C}$, we use $T\mathcal{C}$ to denote its tangent bundle (i.e. a manifold that describes the tangent spaces on the base manifold $\mathcal{C}$) and write $p_{T\mathcal{C}} : T\mathcal{C} \to \mathcal{C}$ to denote the bundle projection, which recovers the corresponding point on $\mathcal{C}$ (i.e. configuration) from a point on $T\mathcal{C}$ (a pair of position and the attached tangent vector). Specifically, suppose $(U, (\mathbf{q}, \mathbf{v}))$ is a (local) chart on $T\mathcal{C}$ on a neighborhood U. Let $\{\frac{\partial}{\partial q_i}, \frac{\partial}{\partial v_i}\}_{i=1}^d$ and $\{\mathrm{d}q^i, \mathrm{d}v^i\}_{i=1}^d$ denote the induced frame field and coframe field on $T\mathcal{C}$ (i.e. the basis vector fields that characterize the tangent spaces and their dual spaces). For $s \in U$, we write s in coordinate as $(\mathbf{q}(q), \mathbf{v}(s))$, if $\sum_{i=1}^d v_i(s) \frac{\partial}{\partial q_i}|_q \in T_q\mathcal{C}$, where $q = p_{T\mathcal{C}}(s) \in \mathcal{C}$. With abuse of notation, we also write $s = (\mathbf{q}, \mathbf{v})$ for short unless clarity is lost. Similarly, a chart $(\tilde{U}, (\mathbf{q}, \mathbf{v}, \mathbf{u}, \mathbf{a}))$ can naturally be constructed on the double tangent bundle $TT\mathcal{C}$, where $\tilde{U} = p_{TT\mathcal{C}}^{-1}(U)$ and $p_{TT\mathcal{C}} : TT\mathcal{C} \to T\mathcal{C}$ is the bundle projection: we write $h = (\mathbf{q}, \mathbf{v}, \mathbf{u}, \mathbf{a}) \in TT\mathcal{C}$ if $\sum_{i=1}^d u_i(h) \frac{\partial}{\partial q_i}|_s + a_i(h) \frac{\partial}{\partial v_i}|_s \in T_s T\mathcal{C}$, where $s = p_{TT\mathcal{C}}(h)$. Under these notations, for a curve $q(t)$ on $\mathcal{C}$, we can write $\ddot{q}(t) \in TT\mathcal{C}$ in coordinate as $(\mathbf{q}(t), \dot{\mathbf{q}}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))$. Finally, we define a geometric object called *affine connection*, which defines how tangent spaces at different points on a manifold are related. Given Christoffel symbols $\Gamma_{i,j}^k$, an affine connection ∇ on $TT\mathcal{C}$ is defined via $\nabla_{\frac{\partial}{\partial s_i}} \frac{\partial}{\partial s_j} = \sum_{k=1}^{2d} \Gamma_{i,j}^k \frac{\partial}{\partial s_k}$, where $\frac{\partial}{\partial s_i} \coloneqq \frac{\partial}{\partial q_i}$ and $\frac{\partial}{\partial s_{i+d}} \coloneqq \frac{\partial}{\partial v_i}$ for $i = 1, \ldots, d$.

Using this new notation, we show that GDSs can be written in a coordinate-free manner in terms of affine connection. Let $T\mathcal{C}$ denote the tangent bundle of $\mathcal{C}$, which is a natural manifold to describe the state space. Precisely, we prove that a GDS on $\mathcal{C}$ can be expressed in terms of a unique, asymmetric affine connection ${}^G\nabla$ that is compatible with a Rieman-

nian metric G (defined by $\mathbf{G}$) on $T\mathcal{C}$. It is important to note that G is defined on $T\mathcal{C}$ *not* the original manifold $\mathcal{C}$. As the metric matrix in a GDS can be velocity dependent, we need a larger manifold.

Theorem 11.5.3. *Let G be a Riemannian metric on $T\mathcal{C}$ such that, for $s = (q, v) \in T\mathcal{C}$, $G(s) = \sum_{i,j} G^v_{ij}(s) dq^i \otimes dq^j + G^a_{ij} dv^i \otimes dv^j$, where $G^v_{ij}(s)$ and G^a_{ij} are symmetric and positive-definite, and $G^v_{ij}(\cdot)$ is differentiable. Then there is a unique affine connection ${}^G\nabla$ that is compatible with G and satisfies, $\Gamma^k_{i,j} = \Gamma^k_{ji}$, $\Gamma^k_{i,j+d} = 0$, and $\Gamma^k_{i+d,j+d} = \Gamma^k_{j+d,i+d}$, for $i, j = 1, \ldots, d$ and $k = 1, \ldots, 2d$. In coordinates, if $G^v_{ij}(\dot{q})$ is identified as $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})$, then $\mathrm{pr}_3({}^G\nabla_{\ddot{q}}\ddot{q})$ can be written as $\mathbf{a}_\mathbf{G} \coloneqq \ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\xi}_\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}})$, where $\mathrm{pr}_3 : (\mathbf{q}, \mathbf{v}, \mathbf{u}, \mathbf{a}) \mapsto \mathbf{u}$ is a projection.*

We call $\mathrm{pr}_3({}^G\nabla_{\dot{q}}\dot{q})$ the *geometric acceleration* of $q(t)$ with respect to ${}^G\nabla$. It is a coordinate-free object, because pr_3 is defined independent of the choice of chart on $\mathcal{C}$. By Theorem 11.5.3, it is clear that a GDS can be written abstractly as

$$\mathrm{pr}_3({}^G\nabla_{\ddot{q}}\ddot{q}) = (\mathrm{pr}_3 \circ G^\sharp \circ F)(s) \tag{11.25}$$

where $F : s \mapsto -d\Phi(s) - B(s)$ defines the covectors due to the potential function and damping, and $G^\sharp : T^*T\mathcal{C} \to T T\mathcal{C}$ denotes the inverse of G. In coordinates, it reads as $\ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\xi}_\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}}) = -\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\nabla_\mathbf{q}\Phi(\mathbf{q}) + \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}})$, which is exactly (13.3).

Extending this result, we present a coordinate-free representation of RMPflow when the leaf-nodes are GDSs.

Theorem 11.5.4. *Suppose $\mathcal{C}$ is related to K leaf-node task spaces by maps $\{\psi_i : \mathcal{C} \to \mathcal{T}_i\}_{i=1}^K$ and the ith task space $\mathcal{T}_i$ has an affine connection ${}^{G_i}\nabla$ on $T\mathcal{T}_i$, as defined in Theorem 11.5.3, and a covector function F_i defined by some potential and damping as described above. Let ${}^G\bar{\nabla} = \sum_{i=1}^K T\psi_i^{*G_i}\nabla$ be the pullback connection, $G = \sum_{i=1}^K T\psi_i^* G_i$ be the pullback metric, and $F = \sum_{i=1}^K T\psi_i^* F_i$ be the pullback covector, where $T\psi_i^* : T^*T\mathcal{T}_i \to$*

$T^*T\mathcal{C}$. Then ${}^G\bar{\nabla}$ is compatible with G, and $\mathrm{pr}_3({}^G\bar{\nabla}_{\dot{q}}\ddot{q}) = (\mathrm{pr}_3 \circ G^\sharp \circ F)(s)$ *can be written as* $\ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q},\dot{\mathbf{q}})^{-1}(\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q},\dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q},\dot{\mathbf{q}})\ddot{\mathbf{q}}) = -\mathbf{G}(\mathbf{q},\dot{\mathbf{q}})^{-1}(\nabla_{\mathbf{q}}\Phi(\mathbf{q}) + \mathbf{B}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}})$. *In particular, if G is velocity-independent, then* ${}^G\bar{\nabla} = {}^G\nabla$.

Theorem 11.5.4 says that the structured GDS $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ can be written abstractly, without coordinates, using the pullback of task-space covectors, metrics, and asymmetric affine connections (that are defined in Theorem 11.5.3). In other words, the recursive calls of `pullback` in the backward pass of RMPflow is indeed performing "pullback" of geometric objects. We can think that the leaf nodes define the asymmetric affine connections, and RMPflow performs `pullback` to pullback those connections onto $\mathcal{C}$ to define ${}^G\bar{\nabla}$. Theorem 11.5.4 also shows, when G is velocity-independent, the pullback of connection and the pullback of metric commutes. In this case, ${}^G\bar{\nabla} = {}^G\nabla$, which is equivalent to the classic Levi-Civita connection of G. The loss of commutativity in general is due to the asymmetric definition of the connection in Theorem 11.5.3, which however is necessary to derive a control law of acceleration, without further referring to higher-order time derivatives.

11.6 Operational Space Control and Geometric Mechanics in View of RMPflow

With the algorithm details and theoretical properties of RMPflow introduced, we now discuss more precisely how RMPflow is connected to and generalizes existing work in geometric mechanics and operational space control.

11.6.1 From Operational Space Control to RMPflow with GDSs

Our study of GDSs (introduced in Section 12.2.2) is motivated by SMSs in geometric mechanics which describe the dynamics used in existing operational space control schemes (cf. Section 11.3). Many formulations of mechanics exist, including Lagrangian mechanics (Taylor, 2005) and the aforementioned Gauss's principle of least constraint (Udwadia and Kalaba, 1996), and they are all equivalent, implicitly sharing the same mathematical structure. But among them, we find that geometric mechanics, which models physical sys-

tems as geodesic flow on Riemannian manifolds, is the most explicit one: it summarizes the system properties arising from the underlying manifold structure compactly, as Riemannian metrics, and connects to the broad mathematical tool set from Riemannian geometry.

These geometry-based insights provide us a way to generalize beyond the previous SMSs studied in (Bullo and Lewis, 2004) and then design GDSs, a family non-classical dynamical systems that, through the use of configuration-and-velocity dependent metrics, more naturally describe behaviors of robots desired for tasks in non-Euclidean spaces.

The proposed generalization preserves several nice features from SMSs to GDSs. As in SMSs, the properties of GDSs are captured by the metric matrix. For example, a GDS like a SMS possesses the natural conservation property of kinematic energy, i.e. it travels along a geodesic defined by $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ when there is no external perturbations due to Φ and $\mathbf{B}$. Note that $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ by definition may only be positive-semidefinite even when the system is non-degenerate; here we allow the geodesic to be defined for a degenerate metric, meaning a curve whose instant length measured by the (degenerate) metric is constant. This geometric feature is an important tool to establish the stability of GDSs in our analysis; We highlight this nice property below, which is a corollary of Proposition 11.5.1. Note that this property also hold for degenerate GDSs provided that differential equations satisfying (11.34) exist.

Corollary 11.6.1. *All GDSs in the form* $(\mathcal{M}, \mathbf{G}, 0, 0)$ *travel on geodesics defined by* $\mathbf{G}$. *That is,* $\dot{K}(\mathbf{x}, \dot{\mathbf{x}}) = 0$, *where* $K(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{2}\dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$.

As we discussed earlier, these generalized metrics induce curvature terms $\boldsymbol{\Xi}_{\mathbf{G}}$ and $\boldsymbol{\xi}_{\mathbf{G}}$ that can be useful to design sensible motions for tasks in non-Euclidean spaces (cf. Section 11.4.6). As we showed GDSs are coordinate-free, these terms and behaviors arise naturally when traveling on geodesics that is defined by configuration-and-velocity dependent metrics. To gain more intuition about these curvature terms, we recall that the curvature term $\boldsymbol{\xi}_{\mathbf{G}}$ in GDSs is related to the Coriolis force in the SMSs. This is not surprising, as from the analysis in Section 11.5.4 we know that $\boldsymbol{\xi}_{\mathbf{G}}$ comes from the Christoffel symbols

of the asymmetric connection in Theorem 11.5.3, just as the Coriolis force comes from the Christoffel symbols of Levi-Civita connection. Recall it is defined as

$$\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \overset{\mathbf{x}}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} - \frac{1}{2}\nabla_{\mathbf{x}}(\dot{\mathbf{x}}^{\top}\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}})$$

Now we show their relationship explicitly below.

Lemma 11.6.1. *Let* $\Gamma_{ijk} = \frac{1}{2}(\partial_{x_k} G_{ij} + \partial_{x_j} G_{ik} - \partial_{x_j} G_{jk})$ *be the Christoffel symbol of the first kind with respect to* $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$*, where the subscript* $_{ij}$ *denotes the* (i, j) *element. Let* $C_{ij} = \sum_{k=1}^{d} \dot{x}_k \Gamma_{ijk}$ *and define* $\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}}) = (C_{ij})_{i,j=1}^{m}$*. Then* $\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$*.*

Proof of Lemma 11.6.1. Suppose $\boldsymbol{\xi}_{\mathbf{G}} = (\xi_i)_{i=1}^{m}$. We can compare the two definitions and verify they are indeed equivalent:

$$\begin{aligned}
\xi_i &= \sum_{j,k=1}^{d} \dot{x}_j \dot{x}_k \partial_{x_j} G_{ik} - \frac{1}{2} \sum_{j,k=1}^{d} \dot{x}_j \dot{x}_k \partial_{x_i} G_{jk} \\
&= \frac{1}{2} \sum_{j,k=1}^{d} \dot{x}_j \dot{x}_k \partial_{x_k} G_{ij} + \frac{1}{2} \sum_{j,k=1}^{d} \dot{x}_j \dot{x}_k \partial_{x_j} G_{ik} - \frac{1}{2} \sum_{j,k=1}^{d} \dot{x}_j \dot{x}_k \partial_{x_i} G_{jk} = (\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}})_i \quad \blacksquare
\end{aligned}$$

Thus, we can think intuitively that GDSs modify the inertia and the Coriolis forces in SMSs so that the dynamical system can preserve a generalized notion of kinematic energy that is no-longer necessarily quadratic in velocity.

Finally, we note that the benefits of using configuration-and-velocity dependent metrics can also be understood from their connection to the weight matrices in least-squared problems. Recall from Section 11.3 that for SMSs, the inertia matrix (which is the same as the metric matrix according to geometric mechanics) forms the importance weight in the least-squared problem. In other words, we can view common operational space control schemes as implicitly combining policies with constant or configuration dependent importance weight matrix in the least-squared sense, which implies certain restriction on the richness of behaviors that it can generate. By contrast, RMPflow allows generally importance weight matrices to depend also on velocity in the least-square problems prescribed by

(11.18), which combines RMPs from the child nodes as a RMP at the parent node in every level of the RMP-tree (cf. Section 11.4). When the polices come from (structured) GDSs, these weight matrices now again are inertia matrices and the geometric properties of GDSs lead to similar stability and convergence properties as their SMS predecessors. Thus, in a sense, we can view RMPflow as generalizing operational space control to consider also configuration-and-velocity dependent weights in policy generation, allowing more flexible trade-offs between different policies.

11.6.2 Relationship between RMPflow and Recursive Newton-Euler Algorithms

For readers familiar with robot dynamics, we remark that the forward-backward policy generation procedure of RMPflow is closely related to the algorithms (Walker and Orin, 1982) for computing forward dynamics (i.e. computing accelerations given forces) based on recursive Newton-Euler algorithm. Here we discuss their relationship.

In a summary, these classic algorithms compute the forward dynamics using following steps:

1. It propagates positions and velocities from the base to the end-effector.
2. It computes the Coriollis force by backward propagating the inverse dynamics of each link under the condition that the acceleration is zero.
3. It computes the (full/upper-triangular/lower-triangular) joint inertia matrix.
4. It solves a linear system of equations to obtain the joint acceleration.

In (Walker and Orin, 1982), they assume a recursive Newton-Euler algorithm (RNE) for inverse dynamics is given, and realize Step 1 and Step 2 above by calling the RNE subroutine. The computation of Step 3 depends on which part of the inertia matrix is computed. In particular, their Method 3 (also called the Composite-Rigid-Body Algorithm in Featherstone, 2008, Chapter 6) computes the upper triangle part of the inertia matrix by a backward propagation from the end-effector to the base.

RMPflow can also be used to compute forward dynamics, when we set the leaf-node policy as the constant inertia system on the body frame of each link and we set the transformation in the RMP-tree as the change of coordinates across of robot links. This works because we showed that when leaf-node policies are GDSs (which cover SMSs of rigid-body dynamics as a special case), the effective dynamics at the root node is the pullback GDS, which in this case is the effective robot dynamics defined by the inertia matrix of each link.

We can use this special case to compare RMPflow with the above procedure. We see that the forward pass of RMPflow is equivalent to Step 1, and the backward pass of RMPflow is equivalent of Step 2 and Step 3, and the final `resolve` operation is equivalent to Step 4.

Despite similarity, the main difference is that RMPflow computes the force and the inertia matrix in a *single* backward pass to exploit shared computations. This change is important, especially, the number of subtasks are large, e.g., in avoiding multiples obstacles. In addition, the design of RMPflow generalizes these classical computational procedures (e.g. designed only for rigid bodies, rotational/prismatic joints) to handle abstract and even non-Euclidean task spaces that have velocity-dependent metrics/inertias. This extension provides a unified framework of different algorithms and results in an expressive class of motion policies.

Finally, we note that the above idea can be slightly modified so that we can also use RMPflow to compute the inverse dynamics. This can be done similarly to the above construction using physical inertia to initialize leaf-node RMPs; but at the end, after the backward pass, we solve for instead the torque as $\boldsymbol{\tau} = \mathbf{M}_r \dot{\mathbf{q}}_d - \mathbf{M}_r \mathbf{f}_r$ where $\dot{\mathbf{q}}_d$ is the desired joint-space acceleration.

11.6.3 Related Approaches to Motion Policy Generation

While here we focus on the special case of RMPflow with GDSs, this family already covers a wide range of reactive policies commonly used in practice. For example, when the task metric is Euclidean (i.e. constant), RMPflow recovers operational space control (and its variants) (Khatib, 1987; Lo, Cheng, and Huang, 2016; Peters et al., 2008; Sentis and Khatib, 2006; Udwadia, 2003). When the task metric is only configuration dependent, RMPflow can be viewed as performing energy shaping to combine multiple SMSs in geometric control (Bullo and Lewis, 2004). Further, RMPflow allows using velocity dependent metrics, generating behaviors all those previous rigid mechanics-based approaches fail to model. We also note that RMPflow can be easily modified to incorporate exogenous time-varying inputs (e.g. forces to realize impedance control (Albu-Schaffer and Hirzinger, 2002) or learned perturbations as in DMPs (Ijspeert et al., 2013)). In computation, the structure of RMPflow in natural-formed RMPs resembles the classical Recursive Newton-Euler algorithm (Featherstone, 2008; Walker and Orin, 1982) (as we just discussed above). Alternatively, the canonical form of RMPflow in (11.19) resembles Gauss's Principle (Peters et al., 2008; Udwadia, 2003), but with a curvature correction $\Xi_{\mathbf{G}}$ on the inertia matrix (suggested by Theorem 11.5.1) to account for velocity dependent metrics. Thus, we can view RMPflow as a natural generalization of these approaches to a broader class of non-Euclidean behaviors.

11.7 Relationship between RMPflow, Factor-Graph, and Sparse Linear Systems

In this section, we discuss the relationship between the computation procedure of RMPflow and algorithms for sparse linear systems. We show that the policy generation process of RMPflow is equivalent to a version of block Gauss elimination. In addition, this identification suggests that the final policy desired in RMPflow can be computed through other solvers of linear systems.

11.7.1 Preliminary: Quadratic Program

Let us refresh some basic properties of linearly constrained quadratic programs (QPs). Consider the abstract QP problem below:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^\top H x + g^\top x \\ s.t. \quad & Ax = b \end{aligned}$$

where H is positive definite, g, b are arbitrary. Without loss of generality, we suppose A is not full rank (otherwise, this is just a linear system). Let λ be the Lagrangian multiplier of the constraint. Then the Lagrangian of this QP can be written as

$$L(x, \lambda) = \frac{1}{2} x^\top H x + g^\top x + \lambda^\top (Ax - b).$$

and the optimal solution to the QP can be solved by the KarushKuhnTucker (KKT) condition

$$\begin{aligned} A^\top \lambda + Hx + g = 0 \\ Ax = b \end{aligned} \quad \Longleftrightarrow \quad \begin{bmatrix} A & 0 \\ H & A^\top \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} b \\ -g \end{bmatrix}$$

11.7.2 The Quadratic Program RMPflow Solves

As mentioned early sections, RMPflow is essentially solving a huge weighted least-squares problem, where the weights are given by the intertia matrices of the task spaces, and the constraints are given by the kinematic relationship of the forward maps. This fact can be observed from the relationship between `pullback` and the weighted least-squares and that only one `resolve` is called at the end.

Let us precisely illustrate this effective QP by considering an RMP-tree with three layers (i.e. depth two). Suppose that we denote the root as `r`, the leafs as `l`, and the others as

$\mathtt{o}$. Then the backward pass of RMPflow solves the QP below

$$\begin{aligned} \min_{a} \quad & \frac{1}{2}(a_{\mathtt{1}} - a_{\mathtt{1}}^d)^\top M_{\mathtt{1}}(a_{\mathtt{1}} - a_{\mathtt{1}}^d) \\ s.t. \quad & a_{\mathtt{1}} = J_{\mathtt{1}} a_{\mathtt{o}} + c_{\mathtt{1}} \\ & a_{\mathtt{o}} = J_{\mathtt{o}} a_{\mathtt{r}} + c_{\mathtt{o}} \end{aligned}$$

where $a_{\mathtt{1}}^d$ is the desired leaf-node accelerations, $M_{\mathtt{1}}$ is a block diagonal matrix given by the inertia matrices of the leaf nodes, $J_{\mathtt{1}}$, $J_{\mathtt{o}}$ are the associated Jacobian matrices, and $c_{\mathtt{1}}$, $c_{\mathtt{o}}$ are the associated curvature terms. Here we adopt a slightly different, abstract notation from the previous sections in this chapter to better highlight the relationship. For example, $a_{\mathtt{1}}$ denotes the collection of all leaf-node accelerations in an RMP-tree, instead of that of a single leaf-node. Similarly, $a_{\mathtt{o}}$ denotes the collections of other intermediate accelerations. Finally, we note that the assumption of three layers does not lose generality. A deeper tree can be handled naturally by introducing extra equality constraints between the root and the leaves.

Define $f_{\mathtt{1}}^d = M_{\mathtt{1}} a_{\mathtt{1}}^d$. We can then rite the objective of this QP as

$$\frac{1}{2}(a_{\mathtt{1}} - a_{\mathtt{1}}^d)^\top M_{\mathtt{1}}(a_{\mathtt{1}} - a_{\mathtt{1}}^d) = \frac{1}{2} a_{\mathtt{1}}^\top M_{\mathtt{1}} a_{\mathtt{1}} - f_{\mathtt{1}}^d a_{\mathtt{1}} + \text{const.}$$

or the full problem as

$$\begin{aligned} \min_{a} \quad & \frac{1}{2} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \end{bmatrix}^\top \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & M_{\mathtt{1}} \end{bmatrix} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ f_{\mathtt{1}}^d \end{bmatrix}^\top \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \end{bmatrix} \\ s.t. \quad & \begin{bmatrix} -J_{\mathtt{o}} & I & 0 \\ 0 & -J_{\mathtt{1}} & I \end{bmatrix} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \end{bmatrix} = \begin{bmatrix} c_{\mathtt{o}} \\ c_{\mathtt{1}} \end{bmatrix} \end{aligned}$$

To solve this QP, we introduce Lagrange multiplier r_1 and r_o for the two constraints, and then we write down the KKT condition of the QP:

$$\begin{bmatrix} -J_o & I & 0 & 0 & 0 \\ 0 & -J_1 & I & 0 & 0 \\ 0 & 0 & 0 & -J_o^\top & 0 \\ 0 & 0 & 0 & I & -J_1^\top \\ 0 & 0 & M_1 & 0 & I \end{bmatrix} \begin{bmatrix} a_r \\ a_o \\ a_1 \\ r_o \\ r_1 \end{bmatrix} = \begin{bmatrix} c_o \\ c_1 \\ 0 \\ 0 \\ f_1^d \end{bmatrix}$$

or equivalently (after exchanging row 3 and 5, and exchanging column 4 and 5)

$$\begin{bmatrix} -J_o & I & 0 & 0 & 0 \\ 0 & -J_1 & I & 0 & 0 \\ 0 & 0 & M_1 & I & 0 \\ 0 & 0 & 0 & -J_1^\top & I \\ 0 & 0 & 0 & 0 & -J_o^\top \end{bmatrix} \begin{bmatrix} a_r \\ a_o \\ a_1 \\ r_1 \\ r_o \end{bmatrix} = \begin{bmatrix} c_o \\ c_1 \\ f_1^d \\ 0 \\ 0 \end{bmatrix} \tag{11.26}$$

From (11.26), it is clear that the Lagrange multiplier r_1 is the residue of the linear equation $M_1 a_1 + r_1 = f_1^d$, and $r_o = J_1^\top r_1$ is the propagated residue.

The linear system (11.26) is block tridiagonal and sparse, with non-empty entries only the edges of the RMP-tree. Therefore, it is possible to use a sparse linear solver to compute all the unknown variables using a time complexity linearly in the number of nodes and edges in the RMP-tree. While a generic solver does provide a solution, it may not be the best idea. We can actually solve this linear system more efficiently by leveraging the fact that in RMPflow we only care about the desired acceleration a_r (the motion policy at the root node) not all the unknown variables.

RMPflow resembles a specific form of back substitution that uses this insight to incompletely solve the linear system. Specially, RMPflow performs the following steps, while grouping shared computations.

1.

$$\begin{bmatrix} -J_{\mathtt{o}} & I & 0 & 0 & 0 \\ 0 & -J_{\mathtt{1}} & I & 0 & 0 \\ 0 & 0 & M_{\mathtt{1}} & I & 0 \\ 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} & 0 \\ 0 & 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} \end{bmatrix} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \\ r_{\mathtt{1}} \\ r_{\mathtt{o}} \end{bmatrix} = \begin{bmatrix} c_{\mathtt{o}} \\ c_{\mathtt{1}} \\ f_{\mathtt{1}}^{d} \\ 0 \\ 0 \end{bmatrix}$$

2.

$$\begin{bmatrix} -J_{\mathtt{o}} & I & 0 & 0 & 0 \\ 0 & -J_{\mathtt{1}} & I & 0 & 0 \\ 0 & 0 & J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} M_{\mathtt{1}} & 0 & 0 \\ 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} & 0 \\ 0 & 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} \end{bmatrix} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \\ r_{\mathtt{1}} \\ r_{\mathtt{o}} \end{bmatrix} = \begin{bmatrix} c_{\mathtt{o}} \\ c_{\mathtt{1}} \\ J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} f_{\mathtt{1}}^{d} \\ 0 \\ 0 \end{bmatrix}$$

3.

$$\begin{bmatrix} -J_{\mathtt{o}} & I & 0 & 0 & 0 \\ 0 & J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} M_{\mathtt{1}} J_{\mathtt{1}} & 0 & 0 & 0 \\ 0 & 0 & J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} M_{\mathtt{1}} & 0 & 0 \\ 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} & 0 \\ 0 & 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} \end{bmatrix} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \\ r_{\mathtt{1}} \\ r_{\mathtt{o}} \end{bmatrix} = \begin{bmatrix} c_{\mathtt{o}} \\ J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} f_{\mathtt{1}}^{d} - J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} M_{\mathtt{1}} c_{\mathtt{1}} \\ J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} f_{\mathtt{1}}^{d} \\ 0 \\ 0 \end{bmatrix}$$

4.

$$\begin{bmatrix} J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} M_{\mathtt{1}} J_{\mathtt{1}} J_{\mathtt{o}} & 0 & 0 & 0 & 0 \\ 0 & J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} M_{\mathtt{1}} J_{\mathtt{1}} & 0 & 0 & 0 \\ 0 & 0 & J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} M_{\mathtt{1}} & 0 & 0 \\ 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} & 0 \\ 0 & 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} \end{bmatrix} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \\ r_{\mathtt{1}} \\ r_{\mathtt{o}} \end{bmatrix} = \begin{bmatrix} J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} (f_{\mathtt{1}}^{d} - M_{\mathtt{1}} (c_{\mathtt{1}} - J_{\mathtt{1}} c_{\mathtt{o}})) \\ J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} (f_{\mathtt{1}}^{d} - M_{\mathtt{1}} c_{\mathtt{1}}) \\ J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top} f_{\mathtt{1}}^{d} \\ 0 \\ 0 \end{bmatrix}$$

In the steps above, we can recognize intermediate variables are $M_{\mathtt{o}}$, $M_{\mathtt{r}}$, $f_{\mathtt{r}}$, $f_{\mathtt{o}}$, and $f_{\mathtt{r}}$ used in RMPflow.

11.7.3 Discussion

We show that RMPflow essentially solves a sparse, block upper tridiagonal system:

$$\begin{bmatrix} -J_{\mathtt{o}} & I & 0 & 0 & 0 \\ 0 & -J_{\mathtt{1}} & I & 0 & 0 \\ 0 & 0 & M_{\mathtt{1}} & I & 0 \\ 0 & 0 & 0 & -J_{\mathtt{1}}^{\top} & I \\ 0 & 0 & 0 & 0 & -J_{\mathtt{o}}^{\top} \end{bmatrix} \begin{bmatrix} a_{\mathtt{r}} \\ a_{\mathtt{o}} \\ a_{\mathtt{1}} \\ r_{\mathtt{1}} \\ r_{\mathtt{o}} \end{bmatrix} = \begin{bmatrix} c_{\mathtt{o}} \\ c_{\mathtt{1}} \\ f_{\mathtt{1}}^{d} \\ 0 \\ 0 \end{bmatrix} \tag{11.26}$$

where all the matrices and vectors are given in the forward pass. Particularly, the matrices $J_{\mathtt{o}}$ and $J_{\mathtt{1}}$ are block sparse, and $M_{\mathtt{1}}$ is block diagonal.

From this identification, we see that RMPflow is not the unique algorithm to achieve this goal. It is also possible to use other existing sparse linear solvers, e.g., a factor-graph solver. Indeed one can identify each row, or a block of rows in (11.26) as a quadratic factor in a factor graph (Dellaert and Kaess, 2017, Chapter 3).

While all linear solvers are admissible here, we remark some important considerations for developing efficient new algorithms. First, because the multiplied matrix $J_{\mathtt{o}}^{\top} J_{\mathtt{1}}^{\top}$ is dense in general, an algorithm for the linear system in (11.26) should not directly create such a matrix in its computation. Instead it needs to retain the sparsity pattern (which is equivalent to performing message passing along the edges of the RMP-tree). This feature can be realized by most standard sparse solvers. In addition, the algorithm should leverage the fact that $a_{\mathtt{r}}$ is only the variable concerned in generating the robot's motion policy. This insight can save computation complexity around by half, and is equivalent to the ordering of variables in solving linear systems. The current design of RMPflow leverages these two properties to achieve efficient computation, but it is nonetheless sequential. Based on the

above insight, parallel solvers to generate the motion policy of RMPflow becomes possible. We consider this an interesting future direction, which is suitable for large-scale or multi-agent systems.

11.8 Experiments

We first perform controlled experiments to study the curvature effects of nonlinear metrics, which is important for stability and collision avoidance. Then we conduct several full-body experiments (video: `https://youtu.be/Fl4WvsXQDzo`) to demonstrate the capabilities of RMPflow on high-DOF manipulation problems in clutter, and implement an integrated vision-and-motion system on two physical robots. Extra details of the RMPs used in this section can found in the Appendix D of the technical report (Cheng et al., 2018c).

11.8.1 Controlled Experiments

1D Example

Let $\mathbf{q} \in \mathbb{R}$. We consider a barrier-type task map $\mathbf{x} = 1/\mathbf{q}$ and define a GDS in (13.3) with $\mathbf{G} = 1$, $\Phi(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^2$, and $\mathbf{B} = (1 + 1/\mathbf{x})$, where $\mathbf{x}_0 > 0$. Using the GDS, we can define an RMP $[-\nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}} - \boldsymbol{\xi}_{\mathbf{G}}, \mathbf{M}]^{\mathbb{R}}$, where $\mathbf{M}$ and $\boldsymbol{\xi}_{\mathbf{G}}$ are defined according to Section 12.2.2. We use this example to study the effects of $\dot{\mathbf{J}}\dot{\mathbf{q}}$ in `pullback` (11.18), where we define $\mathbf{J} = \partial_{\mathbf{q}}\mathbf{x}$. Fig. 11.2 compares the desired behavior (Fig. 11.2a) and the behaviors of correct/incorrect `pullback`. If `pullback` is performed correctly with $\dot{\mathbf{J}}\dot{\mathbf{q}}$, the behavior matches the designed one (Fig. 11.2b). By contrast, if $\dot{\mathbf{J}}\dot{\mathbf{q}}$ is ignored, the observed behavior becomes inconsistent and unstable (Fig. 11.2c). While the instability of neglecting $\dot{\mathbf{J}}\dot{\mathbf{q}}$ can be recovered with a damping $\mathbf{B} = (1 + \frac{\dot{\mathbf{x}}^2}{\mathbf{x}})$ nonlinear in $\dot{\mathbf{x}}$ (suggested in (Lo, Cheng, and Huang, 2016)), the behavior remains inconsistent (Fig. 11.2d).

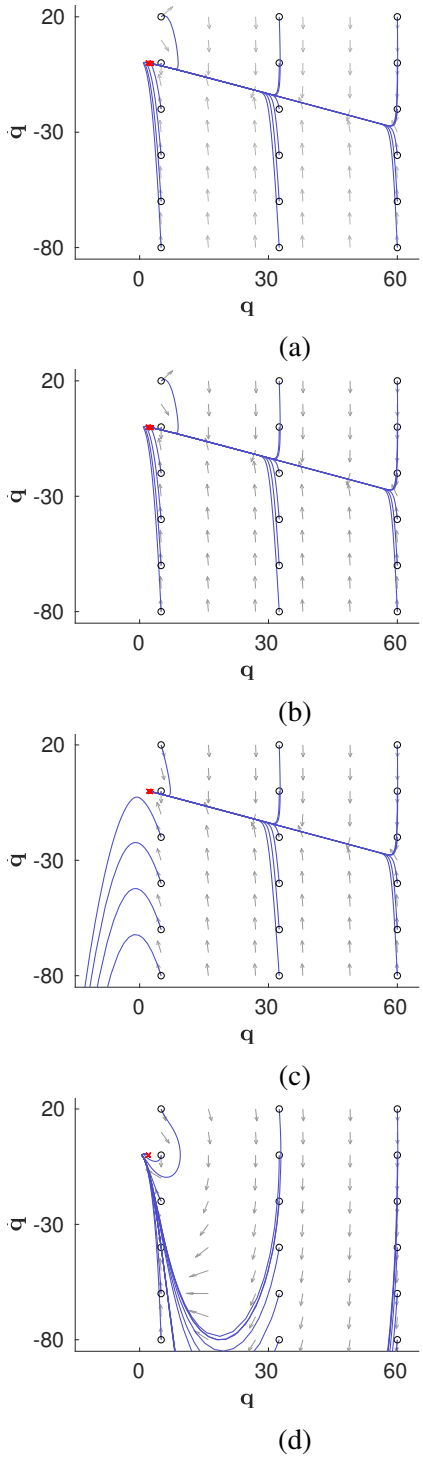


Figure 11.2: Phase portraits (gray) and integral curves (blue; from black circles to red crosses) of 1D example. (a) Desired behavior. (b) With curvature terms. (c) Without curvature terms. (d) Without curvature terms but with nonlinear damping.

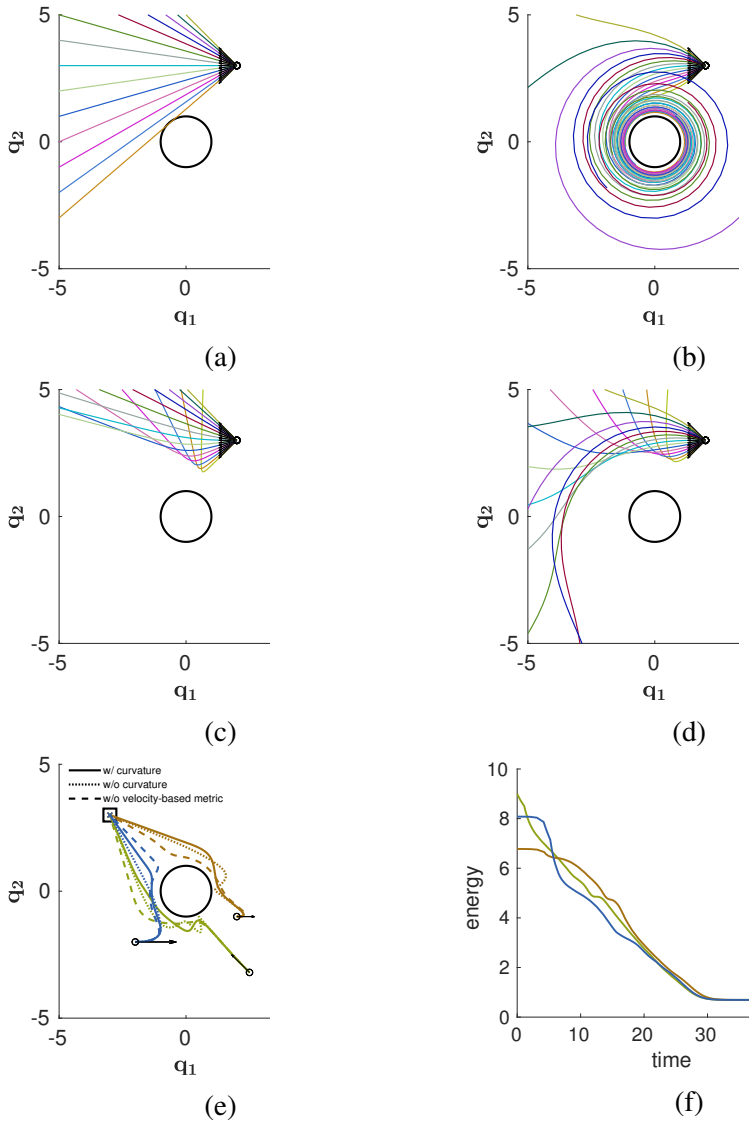


Figure 11.3: 2D example; initial positions (small circle) and velocities (arrows). (a-d) Obstacle (circle) avoidance: (a) w/o curvature terms and w/o potential. (b) w/ curvature terms and w/o potential. (c) w/o curvature terms and w/ potential. (d) w/ curvature terms and w/ potential. (e) Combined obstacle avoidance and goal (square) reaching. (f) The change of Lyapunov function in (11.24) over time along the trajectories in (e).

2D Example

We consider a 2D goal-reaching task with collision avoidance and study the effects of velocity dependent metrics. First, we define an RMP (a GDS as in Section 11.4.6) in $\mathbf{x} = d(\mathbf{q})$ (the 1D task space of the distance to the obstacle). We pick a metric $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = w(\mathbf{x})u(\dot{\mathbf{x}})$, where $w(\mathbf{x}) = 1/\mathbf{x}^4$ increases if the particle is *close* to the obstacle and $u(\dot{\mathbf{x}}) = \epsilon + \min(0, \dot{\mathbf{x}})\dot{\mathbf{x}}$ (where $\epsilon \geq 0$), increases if it moves *towards* the obstacle. As this metric is non-constant, the GDS has curvature terms $\boldsymbol{\Xi}_{\mathbf{G}} = \frac{1}{2}\dot{\mathbf{x}}w(\mathbf{x})\partial_{\dot{\mathbf{x}}}u(\dot{\mathbf{x}})$ and $\boldsymbol{\xi}_{\mathbf{G}} = \frac{1}{2}\dot{\mathbf{x}}^2u(\dot{\mathbf{x}})\partial_{\mathbf{x}}w(\mathbf{x})$. These curvature terms along with $\dot{\mathbf{J}}\dot{\mathbf{q}}$ produce an acceleration that lead to natural obstacle avoidance behavior, coaxing the system toward isocontours of the obstacle (Fig. 11.3b). On the other hand, when the curvature terms are ignored, the particle travels in straight lines with constant velocity (Fig. 11.3a). To define the full collision avoidance RMP, we introduce a barrier-type potential $\Phi(\mathbf{x}) = \frac{1}{2}\alpha w(\mathbf{x})^2$ to create extra repulsive forces, where $\alpha \geq 0$. A comparison of the curvature effects in this setting is shown in Fig. 11.3c and 11.3d (with $\alpha = 1$). Next, we use RMPflow to combine the collision avoidance RMP above (with $\alpha = 0.001$) and an attractor RMP. Let $\mathbf{q}_g$ be the goal. The attractor RMP is a GDS in the task space $\mathbf{y} = \mathbf{q} - \mathbf{q}_g$ with a metric $w(\mathbf{y})\mathbf{I}$, a damping $\eta w(\mathbf{y})\mathbf{I}$, and a potential that is zero at $\mathbf{y} = 0$, where $\eta > 0$ (see Cheng et al., 2018c, Appendix D). Fig. 11.3e shows the trajectories of the combined RMP. The combined non-constant metrics generate a behavior that transitions smoothly towards the goal while heading away from the obstacle. When the curvature terms are ignored (for both RMPs), the trajectories oscillate near the obstacle. In practice, this can result in jittery behavior on manipulators. When the metric is not velocity-based ($\mathbf{G}(\mathbf{x}) = w(\mathbf{x})$) the behavior is less efficient in breaking free from the obstacle to go toward the goal. Finally, we show the change of Lyapunov function (11.24) over time along these trajectories in Fig. 11.3f as verification of our theory.

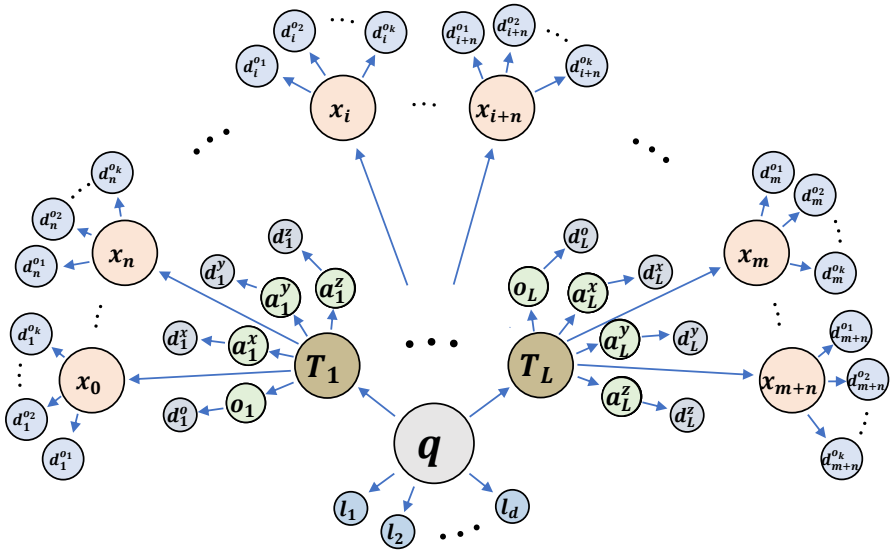


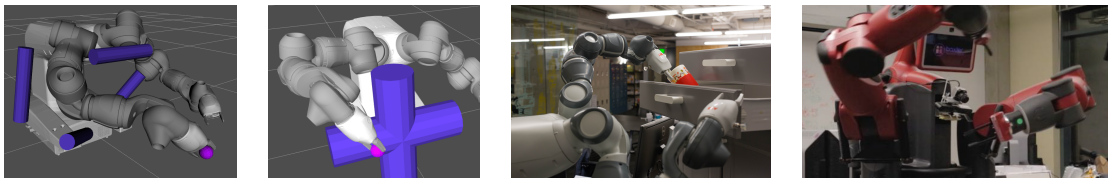
Figure 11.4: This figure depicts the tree of task maps used in the experiments. See Section 11.8.2 for details.

11.8.2 System Experiments

Task map and its Tree Structure

Fig. 11.4 depicts the tree of task maps used in the full-robot experiments. The chosen structure emphasizes potential for parallelization over fully exploiting the recursive nature of the kinematic chain, treating each link frame as just one forward kinematic map step from the configuration space.[9] The configuration space $\mathbf{q}$ is linked to L link frames $\mathbf{T}_1, \ldots, \mathbf{T}_L$ through the robot's forward kinematics (the details of tasks will be described later on for each individual experiment). Each frame has 4 frame element spaces: the origin o_i and each of the axes $\mathbf{a}_i^x, \mathbf{a}_i^y, \mathbf{a}_i^z$, with corresponding distance spaces to targets $d_i^o, d_i^x, d_i^y, d_i^z$ (if they are active). Additionally, there are a number of obstacle control points $\mathbf{x}_j$ distributed across each of the links, each with k associated distance spaces $d_j^{o_1}, \ldots, d_j^{o_k}$, one for each obstacle $o_1, \ldots, o_k$. Finally, for each dimension of the configuration space there's an associated joint

[9] We could possibly have saved some computation by defining the forward kinematic maps recursively as $(\mathbf{T}_{i+1}, \mathbf{q}_{i+1}, \ldots, \mathbf{q}_d) = \psi_i(\mathbf{T}_i, \mathbf{q}_i, \ldots, \mathbf{q}_d)$.



simulated worlds real-world experiments

Figure 11.5: Two of the six simulated worlds in the reaching experiments (left), and the two physical dual-arm platforms in the full system experiment (right).

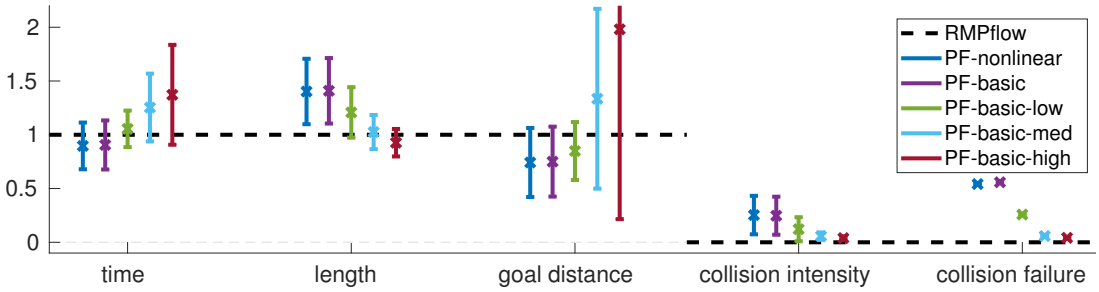


Figure 11.6: Results for reaching experiments. Though some methods achieve a shorter goal distance than RMPflow in successful trials, they end up in collision in most the trials.

limit space $l_1, \ldots, l_d$.

Reaching-through-clutter Experiments

We set up a collection of clutter-filled environments with cylindrical obstacles of varying sizes in simulation as depicted in Fig. 11.5, and tested the performance of RMPflow and two potential field methods on a modeled ABB YuMi robot.

Compared methods:

(i) RMPflow: We implement RMPflow using the RMPs in Section 11.4.6 and detailed in Cheng et al., 2018c, Appendix D. In particular, we place collision-avoidance controllers on distance spaces $s_{ij} = d_j(\mathbf{x}_i)$, where $j = 1, \ldots, m$ indexes the world obstacle o_j and $i = 1, \ldots, n$ indexes the n control point along the robot's body. Each collision-avoidance controller uses a weight function $w_o(\mathbf{x})$ that ranges from 0 when the robot is far from the obstacle to $w_o^{\max} \gg 0$ when the robot is in contact with the obstacle's surface. Similarly, the attractor potential uses a weight function $w_a(\mathbf{x})$ that

ranges from $w_a^{\min}$ far from the target to $w_a^{\max}$ close to the target.

(ii) PF-basic: This variant is a basic implementation of obstacle avoidance potential fields with dynamics shaping. We use the RMP framework to implement this variant by placing collision-avoidance controllers on the same body control points used in RMPflow but with isotropic metrics of the form $\mathbf{G}_o^{\text{basic}}(\mathbf{x}) = w_o^{\max}\mathbf{I}$ for each control point, with $w_o^{\max}$ matching the value RMPflow uses. Similarly, the attractor uses the same attractor potential as RMPflow, but with a constant isotropic metric with the form $\mathbf{G}_a^{\text{basic}}(\mathbf{x}) = w_a^{\max}\mathbf{I}$.

(iii) PF-nonlinear: This variant matches PF-basic in construction, except it uses a *non-linear* isotropic metrics of the form $\mathbf{G}_o^{\text{nlin}}(\mathbf{x}_i) = w_o(\mathbf{x})\mathbf{I}$ and $\mathbf{G}_a^{\text{nlin}}(\mathbf{x}_i) = w_a(\mathbf{x})\mathbf{I}$ for obstacle-avoidance and attraction, respectively, using weight functions matching RMPflow.

A note on curvature terms: PF-basic uses constant metrics, so has no curvature terms; PF-nonlinear has nontrivial curvature terms arising from the spatially varying metrics, but we ignore them here to match common practice from the operational space control literature.

Parameter scaling of PF-basic: Isotropic metrics do not express spacial directionality toward obstacles, and that leads to an inability of the system to effectively trade off the competing controller requirements. That conflict results in more collisions and increased instability. We, therefore, compare PF-basic under these baseline metric weights (matching RMPflow) with variants that incrementally strengthen collision avoidance controllers and C-space postural controllers ($f_{\mathcal{C}}(\mathbf{q}, \dot{\mathbf{q}}) = \gamma_p(\mathbf{q}_0 - \mathbf{q}) - \gamma_d\dot{\mathbf{q}}$) to improve these performance measures in the experiment. We use the following weight scalings (first entry denotes the obstacle metric scalar, and the second entry denotes the C-space metric scalar): "low" $(3, 10)$, "med" $(5, 50)$, and "high" $(10, 100)$.

Environments: We run each of these variants on 6 obstacle environments with 20

randomly sampled target locations each distributed on the opposite side of the obstacle field from the robot. Three of the environments use four smaller obstacles (depicted in panel 3 of Fig. 11.5), and the remaining three environments used two large obstacles (depicted in panel 4 of Fig. 11.5). Each environment used the same 20 targets to avoid implicit sampling bias in target choice.

Performance measures: We report results in Fig. 11.6 in terms of mean and one standard deviation error bars calculated across the 120 trials for each of the following performance measures:[10]

(i) *Time to goal ("time"):* Length of time, in seconds, it takes for the robot to reach a convergence state. This convergence state is either the target, or its best-effort local minimum. If the system never converges, as in the case of many potential field trials for infeasible problems, the trial times out after 5 seconds. This metric measures time-efficiency of the movement.

(ii) *C-space path length ("length"):* This is the total path length $\int \|\dot{\mathbf{q}}\| dt$ of the movement through the configuration space across the trial. This metric measures how economical the movement is. In many of the potential-field variants with lower weights, we see significant fighting among the controllers resulting in highly inefficient extraneous motions.

(iii) *Minimal achievable distance to goal ("goal distance"):* Measures how close, in meters, the system is able to get to the goal with its end-effector.

(iv) *Percent time in collision for colliding trials ("collision intensity"):* Given that a trial has a collision, this metric measures the fraction of time the system is in collision throughout the trial. This metric indicates the intensity of the collision. Low values indicate short grazing collisions while higher values indicate long term obstacle

[10]There is no guarantee of feasibility in planning problems in general, so in all cases, we measure performance relative to the performance of RMPflow, which is empirically stable and near optimal across these problems.

penetration.

(v) *Fraction of trails with collisions ("collision failure"):* Reports the fraction of trials with any collision event. We consider these to be collision-avoidance controller failures.

Discussion: In Fig. 11.6, we see that RMPflow outperforms each of these variants significantly, with some informative trends:

(i) RMPflow never collides, so its collision intensity and collision failure values are 0.

(ii) The other techniques, progressing from no scaling of collision-avoidance and C-space controller weights to substantial scaling, show a profile of substantial collision in the beginning to fewer (but still non-zero) collision events in the end. But we note that improvement in collision-avoidance is achieved at the expense of time-efficiency and the robot's ability to reach the goal (it is too conservative).

(iii) Lower weight scaling of both PF-basic and PF-nonlinear actually achieve some faster times and better goal distances, but that is because the system pushes directly through obstacles, effectively "cheating" during the trial. RMPflow remains highly economical with its best effort reaching behaviors while ensuring the trials remain collision-free.

(iv) Lower weight scalings of PF-basic are highly uneconomical in their motion reflective of their relative instability. As the C-space weights on the posture controllers increase, the stability and economy of motion increase, but, again, at the expense of time-efficiency and optimality of the final reach.

(v) There is little empirical difference between PF-basic and PF-nonlinear indicating that the defining feature separating RMPflow from the potential field techniques is its use of a highly nonlinear metric that explicitly stretches the space in the direction of the obstacle as well as in the direction of the velocity toward the target. Those stretchings

penalize deviations in the stretched directions during combination with other controllers while allowing variation along orthogonal directions. By being more explicit about how controllers should instantaneously trade off with one another, RMPflow is better able to mitigate the otherwise conflicting control signals.

Summary: Isotropic metrics do not effectively convey how each collision and attractor controller should trade off with one another, resulting in a conflict of signals that obscure the intent of each controller making simultaneous collision avoidance, attraction, and posture maintenance more difficult. Increasing the weights of the controllers can improve their effectiveness, but at the expense of decreased overall system performance. The resulting motions are slower and less effective in reaching the goal in spite of more stable behavior and fewer collisions. A key feature of RMPflow is its ability to leverage highly nonlinear metrics that better convey information about how controllers should trade off with one another, while retaining provable stability guarantees. In combination, these features result in efficient and economical obstacle avoidance behavior while reaching toward targets amid clutter.

System Integration for Real-Time Reactive Motion Generation

We demonstrate the integrated vision and motion system on two physical dual arm manipulation platforms: a Baxter robot from Rethink Robotics, and a YuMi robot from ABB. Footage of our fully integrated system (see start of Section 11.8 for the link) depicting tasks such as pick and place amid clutter, reactive manipulation of a cabinet drawers and doors with human interaction, *active* leadthrough with collision controllers running, and pick and place into a cabinet drawer.[11]

This full integrated system, shown in the supplementary video, uses the RMPs described in Section 11.4.6 (detailed in Cheng et al., 2018c, Appendix D) with a slight mod-

[11] We have also run the RMP portion of the system on an ABB IRB120 and a dual arm Kuka manipulation platform with lightweight collaborative arms. Only the two platforms mentioned here, the YuMi and the Baxter, which use the full motion and vision integration, are shown in the video for economy of space.

ification that the curvature terms are ignored. Instead, we maintain theoretical stability by using sufficient damping terms as described in Section 11.8.1 and by operating at slower speeds. Generalization of these RMPs between embodiments was anecdotally pretty consistent, although, as we demonstrate in our experiments, we would expect more empirical deviation at higher speeds. For these manipulation tasks, this early version of the system worked well as demonstrated in the video.

For visual perception, we leveraged consumer depth cameras along with two levels of perceptual feedback:

(i) *Ambient world:* For the Baxter system we create a voxelized representation of the unmodeled ambient world, and use distance fields to focus the collision controllers on just the closest obstacle points surrounding the arms. This methodology is similar in nature to (Kappler et al., 2018), except we found empirically that attending to only the closest point to a skeleton representation resulted in oscillation in concaved regions where distance functions might result in nonsmooth kinks. We mitigate this issue by finding the closest points to a *volume* around each control point, effectively smoothing over points of nondifferentiability in the distance field.

(ii) *Tracked objects:* We use the Dense Articulated Real-time Tracking (DART) system of (Schmidt, Newcombe, and Fox, 2015) to track articulated objects in real time through manipulations. This system is able to track both the robot and environmental objects, such as an articulated cabinet, simultaneously to give accurate measurements of their relative configuration effectively obviating the need for explicit camera-world calibration. As long as the system is initialized in the general region of the object locations (where for the cabinet and the robot, that would mean even up to half a foot of error in translation and a similar scale of error in rotation), the DART optimizer will snap to the right configuration when turned on. DART sends information about object locations to the motion generation, and receives back information about expected joint configurations (priors) from the motion system generating a robust world

representation usable in a number of practical real-world manipulation problems.

Each of our behaviors are decomposed as state machines that use visual feedback to detect transitions, including transitions to reaction states as needed to implement behavioral robustness. Each arm is represented as a separate robot for efficiency, receiving real-time information about other arm's current state enabling coordination. Both arms are programmed simultaneously using a high level language that provides the programmer a unified view of the surrounding world and command of both arms.

11.9 Conclusion

We propose an efficient policy synthesis framework, RMPflow, for generating policies with non-Euclidean behavior, including motion with velocity dependent metrics that are new to the literature. In design, RMPflow is implemented as a computational graph, which can geometrically consistently combine subtask policies into a global policy for the robot. In theory, we provide conditions for stability and show that RMPflow is intrinsically coordinate-free. In the experiments, we demonstrate that RMPflow can generate smooth and natural motion for various tasks, when proper subtask RMPs are specified. Future work is to further relax the requirement on the quality of designing subtask RMPs by introducing learning components into RMPflow for additional flexibility.

11.A Non-holonomic Systems

We can also extend the definition of GDSs to cover non-holonomic constraints. For simplicity, we assume $\boldsymbol{\Xi} \succeq 0$. Without loss of generality, we consider a constraint of the type In this case, we can write

$$\dot{\mathbf{x}} = \mathbf{P}(\mathbf{x})\mathbf{v}(\mathbf{x}) + \mathbf{N}(\mathbf{x})\mathbf{u} \tag{11.27}$$

where $\mathbf{P}(\mathbf{x})$ is a projection matrix that is compatible with $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ and $\mathbf{N}(\mathbf{x}) = \mathbf{I} - \mathbf{P}(\mathbf{x})$, $\mathbf{v}(\mathbf{x})$ is the constraint velocity and $\mathbf{u}$ is the free velocity. We recall a projection $\mathbf{P}$ is compatible with $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ if, for any $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{GPu} = \mathbf{Gu}$. This type of constraint can be used to describe the linearly constraint of the form $\mathbf{L}^\top(\mathbf{x})\dot{\mathbf{x}} = \mathbf{a}$ used in the mechanics literature, or more generally cover the constraints arising from hierarchical control. In the coordinate-free representation, the constraint in (11.27) specifies a *distribution* on the manifold $\mathcal{M}$.

Given that $\mathbf{v}(\mathbf{x})$ in general can be any vector field, here we says a GDS honors the non-holonomic constraint in (11.27), if

$$\mathbf{N}(\mathbf{x})\left(\ddot{\mathbf{x}} + \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})^\dagger(\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}))\right) = -\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})^\dagger(\nabla_{\mathbf{x}}\Phi(\mathbf{x}) + \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}) \tag{11.28}$$

As $\mathbf{N} = \mathbf{I} - \mathbf{P}$ and $\mathbf{P}$ is compatible with $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$, it is not hard to see that if $\mathbf{v}, \Phi, \mathbf{B} = 0$, then (11.28) also travels on the geodesic.

11.B Proofs of RMPflow Analysis

11.B.1 Proof of Theorem 11.5.1

Theorem 11.5.1. *Let the ith child node follow $(\mathcal{N}_i, \mathbf{G}_i, \mathbf{B}_i, \Phi_i)_{\mathcal{S}_i}$ and have coordinate $\mathbf{y}_i$. Let $\mathbf{f}_i = -\boldsymbol{\eta}_{\mathbf{G}_i;\mathcal{S}_i} - \nabla_{\mathbf{y}_i}\Phi_i - \mathbf{B}_i\dot{\mathbf{y}}_i$ and $\mathbf{M}_i = \mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i}$. If $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ of the parent node is given by* `pullback` *with $\{[\mathbf{f}_i, \mathbf{M}_i]^{\mathcal{N}_i}\}_{i=1}^K$ and $\mathbf{M}$ is non-singular, the parent node follows the* pullback structured GDS $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$, *where $\mathbf{G} = \sum_{i=1}^K \mathbf{J}_i^\top \mathbf{G}_i \mathbf{J}_i$, $\mathbf{B} = \sum_{i=1}^K \mathbf{J}_i^\top \mathbf{B}_i \mathbf{J}_i$, $\Phi = \sum_{i=1}^K \Phi_i \circ \mathbf{y}_i$, $\mathcal{S}$ preserves $\mathcal{S}_i$, and $\mathbf{J}_i = \partial_{\mathbf{x}}\mathbf{y}_i$. In other words, the parent node is the RMP $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ where $\mathbf{M} = \sum_{i=1}^K \mathbf{J}_i^\top(\mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i})\mathbf{J}_i$ and*

$$\mathbf{a} = (\mathbf{G} + \boldsymbol{\Xi}_{\mathbf{G}})^\dagger\left(-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} - \nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}}\right)$$

Particularly, if every $\mathbf{G}_i$ is velocity-free and the child nodes are GDSs, the parent node follows $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$.

Proof of Theorem 11.5.1. We will use the non-degeneracy assumption that $\mathbf{G}+\mathbf{\Xi}_{\mathbf{G}}$ (i.e. $\mathbf{M}$ as we will show) is non-singular, so that the differential equation specified by an RMP in normal form or a (structured) GDS is unique. This assumption is made to simplify writing. At the end of the proof, we will show that this assumption only needs to be true at the root node of RMPflow.

The general case We first show the differential equation given by `pullback` is equivalent to the differential equation of pullback structured GDS $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$. Under the non-degeneracy assumption, suppose $\mathcal{S}_i$ factorizes $\mathbf{G}_i$ as $\mathbf{G}_i = \mathbf{L}_i^\top \mathbf{H}_i \mathbf{L}_i$, where $\mathbf{L}_i$ is some Jacobian matrix. On one hand, for `pullback`, because in the child node $\ddot{\mathbf{y}}_i$ satisfies $(\mathbf{G}_i + \mathbf{\Xi}_{\mathbf{G}_i})\ddot{\mathbf{y}}_i = -\boldsymbol{\eta}_{\mathbf{G}_i;\mathcal{S}_i} - \nabla_{\mathbf{y}_i}\Phi_i - \mathbf{B}_i\dot{\mathbf{y}}_i$ (where by definition $\boldsymbol{\eta}_{\mathbf{G}_i;\mathcal{S}_i} = \mathbf{L}_i^\top(\boldsymbol{\xi}_{\mathbf{H}_i} + (\mathbf{H}_i + \mathbf{\Xi}_{\mathbf{H}_i})\dot{\mathbf{L}}_i\dot{\mathbf{y}}_i)$), the `pullback` operator combines the child nodes into the differential equation at the parent node,

$$\mathbf{M}\ddot{\mathbf{x}} = \sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i(\ddot{\mathbf{y}}_i - \dot{\mathbf{J}}_i\dot{\mathbf{x}}) \tag{11.29}$$

where we recall $\mathbf{M} = \sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i$ is given by `pullback`. On the other hand, for $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ with $\mathcal{S}$ preserving $\mathcal{S}_i$, its dynamics satisfy

$$(\mathbf{G} + \mathbf{\Xi}_{\mathbf{G}})\,\ddot{\mathbf{x}} + \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} = -\nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}} \tag{11.30}$$

where $\mathbf{G}$ is factorized by $\mathcal{S}$ into

$$\mathbf{G} = \begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_K \end{bmatrix}^\top \begin{bmatrix} \mathbf{G}_1 & & \\ & \ddots & \\ & & \mathbf{G}_K \end{bmatrix} \begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_K \end{bmatrix} = \begin{bmatrix} \mathbf{L}_1\mathbf{J}_1 \\ \vdots \\ \mathbf{L}_K\mathbf{J}_K \end{bmatrix}^\top \begin{bmatrix} \mathbf{H}_1 & & \\ & \ddots & \\ & & \mathbf{H}_K \end{bmatrix} \begin{bmatrix} \mathbf{L}_1\mathbf{J}_1 \\ \vdots \\ \mathbf{L}_K\mathbf{J}_K \end{bmatrix} =: \bar{\mathbf{J}}^\top \bar{\mathbf{H}} \bar{\mathbf{J}}$$

and the curvature term $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}$ by $\mathcal{S}$ is given as $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} \coloneqq \bar{\mathbf{J}}^\top(\boldsymbol{\xi}_{\bar{\mathbf{H}}} + (\bar{\mathbf{H}} + \boldsymbol{\Xi}_{\bar{\mathbf{H}}})\dot{\bar{\mathbf{J}}}\dot{\mathbf{x}})$.

To prove the general statement, we will show (11.29) and (11.30) are equivalent. First, we introduce a lemma to write $\boldsymbol{\Xi}_{\mathbf{G}}$ in terms of $\boldsymbol{\Xi}_{\mathbf{G}_i}$ (proved later in this section).

Lemma 11.B.1. *Let $\mathcal{M}$ and $\mathcal{N}$ be two manifolds and let $\mathbf{x}$ and $\mathbf{y}(\mathbf{x})$ be the coordinates. Define $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{J}(\mathbf{x})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\mathbf{J}(\mathbf{x})$, where $\mathbf{J}(\mathbf{x}) = \partial_{\mathbf{x}}\mathbf{y}(\mathbf{x})$. Then*

$$\boldsymbol{\Xi}_{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{J}^\top(\mathbf{x})\boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\mathbf{J}(\mathbf{x})$$

Therefore, we see that on the LHSs

$$(\mathbf{G} + \boldsymbol{\Xi}_{\mathbf{G}})\ddot{\mathbf{x}} = \mathbf{M}\ddot{\mathbf{x}}$$

and on the RHSs

$$\begin{aligned}
&\left(\sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i(\ddot{\mathbf{y}}_i - \dot{\mathbf{J}}_i\dot{\mathbf{x}})\right) \\
&= \left(\sum_{i=1}^{K} \mathbf{J}_i^\top(-\boldsymbol{\eta}_{\mathbf{G}_i;\mathcal{S}_i} - \nabla_{\mathbf{y}_i}\Phi_i - \mathbf{B}_i\dot{\mathbf{y}}_i - (\mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i})\dot{\mathbf{J}}_i\dot{\mathbf{x}})\right) \\
&= \left(\sum_{i=1}^{K} \mathbf{J}_i^\top(-\mathbf{L}_i^\top(\boldsymbol{\xi}_{\mathbf{H}_i} + (\mathbf{H}_i + \boldsymbol{\Xi}_{\mathbf{H}_i})\dot{\mathbf{L}}_i\dot{\mathbf{y}}_i) - (\mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i})\dot{\mathbf{J}}_i\dot{\mathbf{x}})\right) + \left(\sum_{i=1}^{K} \mathbf{J}_i^\top(-\nabla_{\mathbf{y}_i}\Phi_i - \mathbf{B}_i\dot{\mathbf{y}}_i)\right) \\
&= \left(\sum_{i=1}^{K} -\bar{\mathbf{J}}_i^\top\boldsymbol{\xi}_{\mathbf{H}_i} - \bar{\mathbf{J}}_i^\top(\mathbf{H}_i + \boldsymbol{\Xi}_{\mathbf{H}_i})(\dot{\mathbf{L}}_i\mathbf{J}_i + \mathbf{L}_i\dot{\mathbf{J}}_i)\dot{\mathbf{x}}\right) - \nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}} \\
&= -\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} - \nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}}
\end{aligned}$$

where the first equality is due to Lemma 11.B.1, the second equality is due to (11.29), and the third equality is due to the definition of structured GDSs. The above derivations show the equivalence between the RHSs and LHSs of (11.29) and (11.30), respectively. Therefore, when the non-degenerate assumption holds, (11.29) and (11.30) are equivalent.

The special case With the closure of structured GDSs proved, we next show the closure

of GDSs under `pullback`, when the metric is only configuration-dependent. That is, we want to show that, when the metric is only configuration-dependent, the choice of structure does not matter. This amounts to show that $\boldsymbol{\xi}_{\mathbf{G}} = \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}$ because by definition $\boldsymbol{\Xi}_i = 0$ and $\boldsymbol{\Xi} = 0$. Below we show how $\boldsymbol{\xi}_{\mathbf{G}}$ is written in terms of $\boldsymbol{\xi}_{\mathbf{G}_i}$ and $\boldsymbol{\Xi}_{\mathbf{G}_i}$ for general metric matrices and specialize it to the configuration-dependent special case (proved later in this section).

Lemma 11.B.2. *Let $\mathcal{M}$ and $\mathcal{N}$ be two manifolds and $\mathbf{x}$ and $\mathbf{y}(\mathbf{x})$ be the coordinates. Suppose $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ is structured as $\mathbf{J}(\mathbf{x})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\mathbf{J}(\mathbf{x})$, where $\mathbf{J}(\mathbf{x}) = \partial_{\mathbf{x}}\mathbf{y}(\mathbf{x})$. Then*

$$\begin{aligned}\boldsymbol{\xi}_{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{J}(\mathbf{x})^\top \Big(\boldsymbol{\xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}) + (\mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) + 2\boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}))\dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}\Big) \\ - \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})^\top \mathbf{J}(\mathbf{x})\dot{\mathbf{x}}\end{aligned}$$

When $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{M}(\mathbf{x})$, $\boldsymbol{\xi}_{\mathbf{M}} = \boldsymbol{\eta}_{\mathbf{M};\mathcal{S}}$ regardless of the structure of $\mathcal{S}$.

By Lemma 11.B.2, we see that structured GDSs are GDSs regardless of the chosen structure when the metric is only configuration dependent. Thus, the statement of the special case follows by combining Lemma 11.B.2 and the previous proof for structured GDSs.

Remarks: Proof of Corollary 11.5.1 We note that the non-degenerate assumption does not need to hold for every nodes in RMPflow but only for the root node. This can be seen from the proof above, where we propagate the LHSs and RHSs *separately*. Therefore, as long as the inertial matrix at the root node is invertible, the differential equation on the configuration space is well defined. ■

Proof of Lemma 11.B.1. Let $\mathbf{b}_i$, $\mathbf{n}_i$, $\mathbf{j}_i$ be the ith column of $\mathbf{M}$, $\mathbf{N}$, and $\mathbf{J}$, respectively. Suppose $\mathcal{M}$ and $\mathcal{N}$ are of m and n dimensions, respectively. By definition of $\boldsymbol{\Xi}_{\mathbf{M}}$,

$$\begin{aligned}2\boldsymbol{\Xi}_{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}}) = \sum_{i=1}^{m} \dot{x}_i \partial_{\dot{\mathbf{x}}} \mathbf{b}_i(\mathbf{x}, \dot{\mathbf{x}}) &= \mathbf{J}(\mathbf{x})^\top \sum_{i=1}^{m} \dot{x}_i \partial_{\dot{\mathbf{x}}} (\mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\mathbf{j}_i(\mathbf{x})) \\ &= \mathbf{J}(\mathbf{x})^\top \left(\sum_{i=1}^{m} \dot{x}_i \partial_{\dot{\mathbf{y}}} (\mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\mathbf{j}_i(\mathbf{x})) \right) \mathbf{J}(\mathbf{x})\end{aligned}$$

$$
\begin{aligned}
&= \mathbf{J}(\mathbf{x})^\top \left(\sum_{j=1}^{n} \partial_{\dot{\mathbf{y}}} \mathbf{n}_j(\mathbf{y}, \dot{\mathbf{y}}) \sum_{i=1}^{m} \dot{x}_i J_{ji}(\mathbf{x}) \right) \mathbf{J}(\mathbf{x}) \\
&= \mathbf{J}(\mathbf{x})^\top \left(\sum_{j=1}^{n} y_j \partial_{\dot{\mathbf{y}}} \mathbf{n}_j(\mathbf{y}, \dot{\mathbf{y}}) \right) \mathbf{J}(\mathbf{x}) \\
&= 2\mathbf{J}(\mathbf{x})^\top \boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}) \mathbf{J}(\mathbf{x}) \qquad \blacksquare
\end{aligned}
$$

Proof of Lemma 11.B.2. Before the proof, we first note a useful identity $\partial_{\mathbf{x}} \dot{\mathbf{y}} = \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})$. This can be derived simply by the definition of the Jacobian matrix $(\partial_{\mathbf{x}} \mathbf{J}(\mathbf{x})\dot{\mathbf{x}})_{ij} = \sum_{k=1}^{m} \dot{x}_k \partial_{x_j} J_{ik} = \sum_{k=1}^{m} \dot{x}_k \partial_{x_j} \partial_{x_k} y_i = \sum_{k=1}^{m} \dot{x}_k \partial_{x_k} J_{ij} = (\dot{\mathbf{J}})_{ij}$.

To prove the lemma, we derive $\boldsymbol{\xi}_{\mathbf{M}}$ by its definition

$$
\begin{aligned}
\boldsymbol{\xi}_{\mathbf{M}} &= \ddot{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} - \frac{1}{2}\nabla_{\mathbf{x}}(\dot{\mathbf{x}}^\top \mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}) \\
&= \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) \mathbf{J}(\mathbf{x})\dot{\mathbf{x}} + \mathbf{J}(\mathbf{x})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{J}(\mathbf{x})^\top \ddot{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}) \mathbf{J}(\mathbf{x})\dot{\mathbf{x}} \\
&\quad - \frac{1}{2}\nabla_{\mathbf{x}}(\dot{\mathbf{x}}^\top \mathbf{J}(\mathbf{x})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) \mathbf{J}(\mathbf{x})\dot{\mathbf{x}}) \\
&= \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} + \mathbf{J}(\mathbf{x})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{J}(\mathbf{x})^\top \ddot{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} - \frac{1}{2}\nabla_{\mathbf{x}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) \\
&= \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} + \mathbf{J}(\mathbf{x})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{J}(\mathbf{x})^\top \ddot{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} \\
&\quad - \frac{1}{2}\mathbf{J}(\mathbf{x})^\top \nabla_{\mathbf{y}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) - \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} - \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})^\top \mathbf{J}(\mathbf{x})\dot{\mathbf{x}} \\
&= \mathbf{J}(\mathbf{x})^\top (\mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \ddot{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}) \mathbf{J}(\mathbf{x})\dot{\mathbf{x}} - \frac{1}{2}\nabla_{\mathbf{y}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}})) - \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})^\top \mathbf{J}(\mathbf{x})\dot{\mathbf{x}}
\end{aligned}
$$

In the second to the last equality above, we use $\partial_{\mathbf{x}} \dot{\mathbf{y}} = \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})$ and derive

$$
\begin{aligned}
\frac{1}{2}\nabla_{\mathbf{x}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) &= \frac{1}{2}\mathbf{J}^\top \nabla_{\mathbf{y}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) + \frac{1}{2}\nabla_{\mathbf{x}}(\dot{\mathbf{y}}) \nabla_{\dot{\mathbf{y}}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) \\
&= \frac{1}{2}\mathbf{J}^\top \nabla_{\mathbf{y}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) + \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} + \frac{1}{2}\dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \nabla_{\dot{\mathbf{y}}}(\mathbf{z}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{z}})|_{\mathbf{z}=\dot{\mathbf{y}}} \\
&= \frac{1}{2}\mathbf{J}^\top \nabla_{\mathbf{y}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) + \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} + \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})^\top \mathbf{J}(\mathbf{x})\dot{\mathbf{x}}
\end{aligned}
$$

as $\frac{1}{2}\partial_{\dot{\mathbf{y}}}(\mathbf{z}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{z}})|_{\mathbf{z}=\dot{\mathbf{y}}} = \frac{1}{2}\dot{\mathbf{y}}^\top \left(\sum_{i=1}^{n} \dot{y}_i \partial_{\dot{\mathbf{y}}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}})\right) = \dot{\mathbf{y}}^\top \boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})$, where $\mathbf{n}_i$ is the ith column of $\mathbf{N}$.

To further simplify the expression, we note that by $\partial_{\mathbf{x}}\dot{\mathbf{y}} = \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})$ we have

$$\begin{aligned}
\overset{\mathbf{x}}{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} &= \sum_{i=1}^{n} \dot{y}_i \partial_{\mathbf{x}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{x}} \\
&= \sum_{i=1}^{n} \dot{y}_i (\partial_{\mathbf{y}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}})\mathbf{J}(\mathbf{x})\dot{\mathbf{x}} + \partial_{\dot{\mathbf{y}}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}})\partial_{\mathbf{x}}(\dot{\mathbf{y}})\dot{\mathbf{x}}) \\
&= \sum_{i=1}^{n} \dot{y}_i \partial_{\mathbf{y}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} + \dot{y}_i \partial_{\dot{\mathbf{y}}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} \\
&= \left(\sum_{i=1}^{n} \dot{y}_i \partial_{\mathbf{y}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}}) \right) \dot{\mathbf{y}} + \left(\sum_{i=1}^{n} \dot{y}_i \partial_{\dot{\mathbf{y}}} \mathbf{n}_i(\mathbf{y}, \dot{\mathbf{y}}) \right) \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} \\
&= \overset{\mathbf{y}}{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} + 2\boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}
\end{aligned}$$

Combining these two equalities, we can write

$$\begin{aligned}
\boldsymbol{\xi}_{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{J}(\mathbf{x})^\top \Bigg(&\overset{\mathbf{y}}{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} - \frac{1}{2}\nabla_{\mathbf{y}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}}) + (\mathbf{N}(\mathbf{y}, \dot{\mathbf{y}}) + 2\boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}))\dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} \Bigg) \\
&- \dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})^\top \boldsymbol{\Xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})^\top \mathbf{J}(\mathbf{x})\dot{\mathbf{x}}
\end{aligned}$$

Substituting the definition of $\boldsymbol{\xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}) = \overset{\mathbf{y}}{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}} - \frac{1}{2}\nabla_{\mathbf{y}}(\dot{\mathbf{y}}^\top \mathbf{N}(\mathbf{y}, \dot{\mathbf{y}})\dot{\mathbf{y}})$ proves the general statement.

In the special case, $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{M}(\mathbf{x})$ (which implies $\boldsymbol{\Xi}_{\mathbf{M}} = 0$),

$$\boldsymbol{\xi}_{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{J}(\mathbf{x})^\top \Big(\boldsymbol{\xi}_{\mathbf{N}}(\mathbf{y}, \dot{\mathbf{y}}) + \mathbf{N}(\mathbf{y})\dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} \Big)$$

We show this expression is equal to $\boldsymbol{\eta}_{\mathbf{M};\mathcal{S}}$ regardless of the structure $\mathcal{S}$. This can be seen from the follows: If further $\mathbf{N}(\mathbf{y}) = \mathbf{L}(\mathbf{y})^\top \mathbf{C}(\mathbf{z})\mathbf{L}(\mathbf{y})$ and $\mathbf{M}$ is structured as $(\mathbf{LJ})^\top \mathbf{C}(\mathbf{LJ})$ from some Jacobian matrix $\mathbf{L}(\mathbf{y}) = \partial_{\mathbf{y}}\mathbf{z}$, we can write

$$\begin{aligned}
\boldsymbol{\eta}_{\mathbf{M};\mathcal{S}} &= \mathbf{J}^\top \mathbf{L}^\top (\boldsymbol{\xi}_{\bar{\mathbf{C}}} + \mathbf{C}\frac{d(\mathbf{LJ})}{dt}\dot{\mathbf{x}}) \\
&= \mathbf{J}^\top (\mathbf{L}^\top \boldsymbol{\xi}_{\bar{\mathbf{C}}} + \mathbf{L}^\top \mathbf{C}(\dot{\mathbf{L}}\mathbf{J} + \mathbf{L}\dot{\mathbf{J}})\dot{\mathbf{x}})
\end{aligned}$$

$$= \mathbf{J}^\top \left(\mathbf{L}^\top (\boldsymbol{\xi}_{\bar{\mathbf{C}}} + \mathbf{C}\dot{\mathbf{L}}\dot{\mathbf{y}}) + \mathbf{L}^\top \mathbf{C}\mathbf{L}\dot{\mathbf{J}}\dot{\mathbf{x}} \right)$$
$$= \mathbf{J}^\top \left(\boldsymbol{\xi}_{\mathbf{N}} + \mathbf{N}\dot{\mathbf{J}}\dot{\mathbf{x}} \right) = \boldsymbol{\xi}_{\mathbf{M}} \qquad \blacksquare$$

11.B.2 Proof of Proposition 11.5.1

Proposition 11.5.1. *For* $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$*, it holds that* $\dot{\mathbf{V}}(\mathbf{q}, \dot{\mathbf{q}}) = -\dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$.

Proof of Proposition 11.5.1. Let $K(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$. Its time derivative can be written as

$$\begin{aligned}
\frac{d}{dt}K(\mathbf{q}, \dot{\mathbf{q}}) &= \dot{\mathbf{q}}^\top \left(\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} + \frac{1}{2}(\frac{d}{dt}\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}))\dot{\mathbf{q}} \right) \\
&= \dot{\mathbf{q}}^\top \left(\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} + \frac{1}{2}\sum_{i=1}^{d} \dot{q}_i \frac{d}{dt}\mathbf{g}_i(\mathbf{q}, \dot{\mathbf{q}}) \right) \\
&= \dot{\mathbf{q}}^\top \left(\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} + \frac{1}{2}\sum_{i=1}^{d} \dot{q}_i \partial_{\mathbf{q}}\mathbf{g}_i(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \frac{1}{2}\sum_{i=1}^{d} \dot{q}_i \partial_{\dot{\mathbf{q}}}\mathbf{g}_i(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} \right) \\
&= \dot{\mathbf{q}}^\top \left((\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}))\ddot{\mathbf{q}} + \frac{1}{2}\overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \right)
\end{aligned}$$

where we recall $\mathbf{G}$ is symmetric and $\overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}}) \coloneqq [\partial_{\mathbf{q}}\mathbf{g}_i(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}]_{i=1}^{d}$. Therefore, by definition $(\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}))\ddot{\mathbf{q}} = (-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q}, \dot{\mathbf{q}}) - \nabla_{\mathbf{q}}\Phi(\mathbf{q}) - \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}(\mathbf{q}, \dot{\mathbf{q}}))$, we can derive

$$\begin{aligned}
\frac{d}{dt}\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) &= \frac{d}{dt}K(\mathbf{q}, \dot{\mathbf{q}}) + \dot{\mathbf{q}}^\top \nabla_{\mathbf{q}}\Phi(\mathbf{q}) \\
&= \dot{\mathbf{q}}^\top \left(-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q}, \dot{\mathbf{q}}) - \nabla_{\mathbf{q}}\Phi(\mathbf{q}) - \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{x}} + \frac{1}{2}\overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \nabla_{\mathbf{q}}\Phi(\mathbf{q}) \right) \\
&= -\dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \dot{\mathbf{q}}^\top \left(-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q}, \dot{\mathbf{q}}) + \frac{1}{2}\overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \right)
\end{aligned}$$

To finish the proof, we use two lemmas below.

Lemma 11.B.3. $\frac{1}{2}\dot{\mathbf{q}}^\top \overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \dot{\mathbf{q}}^\top \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})$.

Proof of Lemma 11.B.3. This can be shown by definition:

$$\begin{aligned}
\dot{\mathbf{q}}^\top \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) &= \dot{\mathbf{q}}^\top \left(\overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \frac{1}{2}\nabla_{\mathbf{q}}(\dot{\mathbf{q}}^\top \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}) \right) \\
&= \sum_{k=1}^{d} \dot{q}_k \left(\sum_{i,j=1}^{d} \dot{q}_i \dot{q}_j \partial_{q_j} G_{k,i} - \frac{1}{2} \sum_{i,j=1}^{d} \dot{q}_i \dot{q}_j \partial_{q_k} G_{i,j} \right) \\
&= \sum_{i,j,k=1}^{d} \dot{q}_i \dot{q}_j \dot{q}_k \partial_{q_j} G_{k,i} - \frac{1}{2} \sum_{i,j,k=1}^{d} \dot{q}_i \dot{q}_j \dot{q}_k \partial_{q_k} G_{i,j} \\
&= \sum_{i,j,k=1}^{d} \dot{q}_i \dot{q}_j \dot{q}_k \partial_{q_k} G_{j,i} - \frac{1}{2} \sum_{i,j,k=1}^{d} \dot{q}_i \dot{q}_j \dot{q}_k \partial_{q_k} G_{i,j} \\
&= \frac{1}{2} \sum_{i,j,k=1}^{d} \dot{q}_i \dot{q}_j \dot{q}_k \partial_{q_j} G_{k,i} = \frac{1}{2} \dot{\mathbf{q}}^\top \overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}
\end{aligned}$$

where for the second to the last equality we use the symmetry $G_{i,j} = G_{j,i}$. ■

Using Lemma 11.B.3, we can show another equality.

Lemma 11.B.4. *For all structure $\mathcal{S}$,* $\dot{\mathbf{q}}^\top \left(-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q}, \dot{\mathbf{q}}) + \frac{1}{2} \overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \right) = 0$

Proof of Lemma 11.B.4. This can be seen from Lemma 11.B.2. Suppose $\mathcal{S}$ factorizes $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}(\mathbf{q})^\top \mathbf{H}(\mathbf{x}, \dot{\mathbf{x}})\mathbf{J}(\mathbf{q})$ where $\mathbf{J}(\mathbf{q}) = \partial_{\mathbf{q}}\mathbf{x}$. By Lemma 11.B.2, we know

$$\boldsymbol{\xi}_{\mathbf{G}} = \mathbf{J}^\top \left(\boldsymbol{\xi}_{\mathbf{H}} + (\mathbf{H} + 2\boldsymbol{\Xi}_{\mathbf{H}})\dot{\mathbf{J}}\dot{\mathbf{x}} \right) - \dot{\mathbf{J}}^\top \boldsymbol{\Xi}_{\mathbf{H}}^\top \mathbf{J}\dot{\mathbf{x}}$$

On the other hand, by definition, we have $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} \coloneqq \mathbf{J}^\top(\boldsymbol{\xi}_{\mathbf{H}} + (\mathbf{H} + \boldsymbol{\Xi}_{\mathbf{H}})\dot{\mathbf{J}}\dot{\mathbf{x}})$. Therefore, by comparing the two, we can derive,

$$\dot{\mathbf{q}}^\top \boldsymbol{\xi}_{\mathbf{G}} = \dot{\mathbf{q}}^\top \left(\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} + \mathbf{J}^\top \boldsymbol{\Xi}_{\mathbf{H}} \dot{\mathbf{J}}\dot{\mathbf{q}} - \dot{\mathbf{J}}^\top \boldsymbol{\Xi}_{\mathbf{H}}^\top \mathbf{J}\dot{\mathbf{q}} \right) = \dot{\mathbf{q}}^\top \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}$$

Combing the above equality and Lemma 11.B.3 proves the equality. ■

Finally, we use Lemma 11.B.4 and the previous result and conclude

$$\frac{d}{dt}\mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) = -\dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \dot{\mathbf{q}}^\top \left(-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q},\dot{\mathbf{q}}) + \frac{1}{2}\overset{\mathbf{q}}{\dot{\mathbf{G}}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\right) = -\dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} \blacksquare$$

11.B.3 Proof of Theorem 11.5.2

Theorem 11.5.2. *Suppose every leaf node is a GDS with a metric matrix in the form* $\mathbf{R}(\mathbf{x}) + \mathbf{L}(\mathbf{x})^\top \mathbf{D}(\mathbf{x},\dot{\mathbf{x}})\mathbf{L}(\mathbf{x})$ *for differentiable functions* $\mathbf{R}$*,* $\mathbf{L}$*, and* $\mathbf{D}$ *satisfying* $\mathbf{R}(\mathbf{x}) \succeq 0$*,* $\mathbf{D}(\mathbf{x},\dot{\mathbf{x}}) = \operatorname{diag}((d_i(\mathbf{x},\dot{y}_i))_{i=1}^n) \succeq 0$*, and* $\dot{y}_i \partial_{\dot{y}_i} d_i(\mathbf{x},\dot{y}_i) \geq 0$*, where* $\mathbf{x}$ *is the coordinate of the leaf-node manifold and* $\dot{\mathbf{y}} = \mathbf{L}\dot{\mathbf{x}} \in \mathbb{R}^n$*. It holds* $\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q},\dot{\mathbf{q}}) \succeq 0$*. If further* $\mathbf{G}(\mathbf{q},\dot{\mathbf{q}}), \mathbf{B}(\mathbf{q},\dot{\mathbf{q}}) \succ 0$*, then* $\mathbf{M} \in \mathbb{R}^{d\times d}_{++}$*, and the global RMP generated by RMPflow converges to the forward invariant set* $\mathcal{C}_\infty$ *in Corollary 11.5.2.*

Proof. Let $\mathbf{A}(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{R}(\mathbf{x}) + \mathbf{L}(\mathbf{x})^\top \mathbf{D}(\mathbf{x},\dot{\mathbf{x}})\mathbf{L}(\mathbf{x})$. The proof of the theorem is straightforward, if we show that $\boldsymbol{\Xi}_{\mathbf{A}}(\mathbf{x},\dot{\mathbf{x}}) \succeq 0$. To see this, suppose $\mathbf{L} = \mathbb{R}^{n\times m}$. Let $\boldsymbol{\omega}_j^\top$ be the jth row $\mathbf{L}$, respectively. By definition of $\boldsymbol{\Xi}_{\mathbf{A}}(\mathbf{x},\dot{\mathbf{x}})$ we can write

$$\begin{aligned}
\boldsymbol{\Xi}_{\mathbf{A}}(\mathbf{x},\dot{\mathbf{x}}) &= \frac{1}{2}\sum_{i=1}^m \dot{x}_i \partial_{\dot{\mathbf{x}}} \mathbf{a}_i(\mathbf{x},\dot{\mathbf{x}}) \\
&= \frac{1}{2}\mathbf{L}(\mathbf{x})^\top \sum_{i=1}^m \dot{x}_i \partial_{\dot{\mathbf{x}}} (\mathbf{D}(\mathbf{x},\dot{\mathbf{x}})\mathbf{l}_i(\mathbf{x})) \\
&= \frac{1}{2}\mathbf{L}(\mathbf{x})^\top \sum_{i=1}^m \sum_{j=1}^n \dot{x}_i \partial_{\dot{\mathbf{x}}} (d_j(\mathbf{x},\dot{y}_j) L_{ji}(\mathbf{x})\mathbf{e}_j) \\
&= \frac{1}{2}\mathbf{L}(\mathbf{x})^\top \sum_{j=1}^n \left(\sum_{i=1}^m L_{ji}(\mathbf{x})\dot{x}_i\right) \partial_{\dot{y}_j} d_j(\mathbf{x},\dot{y}_j)\mathbf{e}_j \boldsymbol{\omega}_j^\top \\
&= \frac{1}{2}\mathbf{L}(\mathbf{x})^\top \sum_{j=1}^n \dot{y}_j \partial_{\dot{y}_j} d_j(\mathbf{x},\dot{y}_j)\mathbf{e}_j \boldsymbol{\omega}_j^\top \\
&= \mathbf{L}(\mathbf{x})^\top \boldsymbol{\Xi}_{\mathbf{D}}(\mathbf{x},\dot{\mathbf{x}})\mathbf{L}(\mathbf{x})
\end{aligned}$$

where $\mathbf{e}_j$ the jth canonical basis and $\boldsymbol{\Xi}_{\mathbf{D}}(\mathbf{x},\dot{\mathbf{x}}) = \frac{1}{2}\operatorname{diag}((\partial_{\dot{y}_i} d_i(\mathbf{x},\dot{y}_i))_{i=1}^n)$. Therefore, under the assumption that $\partial_{\dot{y}_i} d_i(\mathbf{x},\dot{y}_i) \geq 0$, $\boldsymbol{\Xi}_{\mathbf{A}}(\mathbf{x},\dot{\mathbf{x}}) \succeq 0$. This further implies $\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q},\dot{\mathbf{q}}) \succeq 0$

by Theorem 11.5.1.

The stability of the entire system follows naturally from the rule of `pullback`, which ensures that $\mathbf{M}_r(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{M}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$ given that the leaf-node condition is satisfied. Consequently, the condition in Corollary 11.5.2 holds and the convergence to $\mathcal{C}_\infty$ is guaranteed. ■

11.B.4 Notation for Coordinate-Free Analysis

We introduce some extra notations for the coordinate-free analysis. Let $p_{T\mathcal{C}} : T\mathcal{C} \to \mathcal{C}$ be the bundle projection. Suppose $(U, (\mathbf{q}, \mathbf{v}))$ is a (local) chart on $T\mathcal{C}$. Let $\{\frac{\partial}{\partial q_i}, \frac{\partial}{\partial v_i}\}_{i=1}^d$ and $\{\mathrm{d}q^i, \mathrm{d}v^i\}_{i=1}^d$ denote the induced frame field and coframe field on $T\mathcal{C}$. For $s \in U$, we write s in coordinate as $(\mathbf{q}(q), \mathbf{v}(s))$, if $\sum_{i=1}^d v_i(s) \frac{\partial}{\partial q_i}|_q \in T_q\mathcal{C}$, where $q = p_{T\mathcal{C}}(s) \in \mathcal{C}$. With abuse of notation, we also write $s = (\mathbf{q}, \mathbf{v})$ for short unless clarity is lost. Similarly, a chart $(\tilde{U}, (\mathbf{q}, \mathbf{v}, \mathbf{u}, \mathbf{a}))$ can naturally be constructed on the double tangent bundle $TT\mathcal{C}$, where $\tilde{U} = p_{TT\mathcal{C}}^{-1}(U)$ and $p_{TT\mathcal{C}} : TT\mathcal{C} \to T\mathcal{C}$ is the bundle projection: we write $h = (\mathbf{q}, \mathbf{v}, \mathbf{u}, \mathbf{a}) \in TT\mathcal{C}$ if $\sum_{i=1}^d u_i(h) \frac{\partial}{\partial q_i}|_s + a_i(h) \frac{\partial}{\partial v_i}|_s \in T_s T\mathcal{C}$, where $s = p_{TT\mathcal{C}}(h)$. Under these notations, for a curve $q(t)$ on $\mathcal{C}$, we can write $\ddot{q}(t) \in TT\mathcal{C}$ in coordinate as $(\mathbf{q}(t), \dot{\mathbf{q}}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))$. Finally, given Christoffel symbols $\Gamma_{i,j}^k$, an affine connection ∇ on $TT\mathcal{C}$ is defined via $\nabla_{\frac{\partial}{\partial s_i}} \frac{\partial}{\partial s_j} = \sum_{k=1}^{2d} \Gamma_{i,j}^k \frac{\partial}{\partial s_k}$, where $\frac{\partial}{\partial s_i} \coloneqq \frac{\partial}{\partial q_i}$ and $\frac{\partial}{\partial s_{i+d}} \coloneqq \frac{\partial}{\partial v_i}$ for $i = 1, \ldots, d$.

11.B.5 Proof of Theorem 11.5.3

Theorem 11.5.3. *Let G be a Riemannian metric on $T\mathcal{C}$ such that, for $s = (q, v) \in T\mathcal{C}$, $G(s) = \sum_{i,j} G_{ij}^v(s) dq^i \otimes dq^j + G_{ij}^a dv^i \otimes dv^j$, where $G_{ij}^v(s)$ and G_{ij}^a are symmetric and positive-definite, and $G_{ij}^v(\cdot)$ is differentiable. Then there is a unique affine connection ${}^G\nabla$ that is compatible with G and satisfies, $\Gamma_{i,j}^k = \Gamma_{ji}^k$, $\Gamma_{i,j+d}^k = 0$, and $\Gamma_{i+d,j+d}^k = \Gamma_{j+d,i+d}^k$, for $i, j = 1, \ldots, d$ and $k = 1, \ldots, 2d$. In coordinates, if $G_{ij}^v(\dot{q})$ is identified as $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})$, then $\mathrm{pr}_3({}^G\nabla_{\ddot{q}}\ddot{q})$ can be written as $\mathbf{a}_{\mathbf{G}} \coloneqq \ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}})$, where*

$\mathrm{pr}_3 : (\mathbf{q}, \mathbf{v}, \mathbf{u}, \mathbf{a}) \mapsto \mathbf{u}$ *is a projection.*

Proof of Theorem 11.5.3. We first show ${}^G\nabla$ is unique, if it exists. That is, there is at most one affine connection that is compatible with the given the Riemannian metric G and satisfies for $i, j = 1, \ldots, d$ and $k = 1, \ldots, 2d$

$$\Gamma^k_{i,j} = \Gamma^k_{ji}, \qquad \Gamma^k_{i,j+d} = 0, \qquad \Gamma^k_{i+d,j+d} = \Gamma^k_{j+d,i+d},$$

Importantly, we note that this definition is coordinate-free, independent of the choice of chart on $\mathcal{C}$.

The uniqueness is easy to see. As G is non-degenerate by definition, we recall there is an unique Levi-Civita connection, which is compatible with G and satisfies the symmetric condition

$$\Gamma^k_{i,j} = \Gamma^k_{j,i}, \qquad \text{for } i, j = 1, \ldots, 2d$$

Comparing our asymmetric condition and the symmetric condition of the Levi-Civita connection, we see that number of the linearly independent constraints are the same; therefore if there is a solution to the required asymmetric affine connection, then it is unique.

Next we show such a solution exists. We consider the candidate Christoffel symbols below and show that they satisfy the requirements: Consider an *arbitrary* choice of chart on $\mathcal{C}$. For $i, j, k = 1, \ldots, d$,

$$\Gamma^k_{i,j} = \frac{1}{2} \sum_{l=1}^{d} G^{v\sharp}_{k,l} (\partial_{q_j} G^v_{l,i} + \partial_{q_i} G^v_{l,j} - \partial_{q_l} G^v_{i,j})$$

$$\Gamma^k_{i,j+d} = 0, \quad \Gamma^k_{i+d,j} = \frac{1}{2} \sum_{l=1}^{d} G^{v\sharp}_{k,l} (\partial_{v_i} G^v_{l,j}), \quad \Gamma^k_{i+d,j+d} = 0$$

$$\Gamma^{k+d}_{i,j} = 0, \quad \Gamma^{k+d}_{i,j+d} = 0, \quad \Gamma^{k+d}_{i+d,j} = 0, \quad \Gamma^{k+d}_{i+d,j+d} = 0$$

where $G^{v\sharp}$ denotes the inverse of G^v, i.e. $\sum_{k=1}^{d} G^{v\sharp}_{i,k} G^v_{k,j} = \delta_{i,j}$. Note we choose not to

adopt the Einstein summation notation, so the sparse pattern of the proposed Christoffel symbols are clear.

It is clear that the above candidate Christoffel symbols satisfies the asymmetric condition. Therefore, to show it is a solution, we only need to show such choice is compatible with G. Equivalently, it means for arbitrary smooth sections of $TT\mathcal{C}$, $X = \sum_{i=1}^{2d} X_i \frac{\partial}{\partial s_i}$, $Y = \sum_{i=1}^{2d} Y_i \frac{\partial}{\partial s_i}$, $Z = \sum_{i=1}^{2d} Z_i \frac{\partial}{\partial s_i}$, it holds[12]

$$ {}^{G}\nabla_Z G(X,Y) = G({}^{G}\nabla_Z X, Y) + G(X, {}^{G}\nabla_Z Y) \tag{11.31} $$

To verify (11.31), we first write out ${}^{G}\nabla_Z X$ using the chosen Christoffel symbols:

$$ \begin{aligned} {}^{G}\nabla_Z X &= \sum_{k=1}^{2d} \left({}^{G}\nabla_Z X_k + \sum_{i,j=1}^{2d} \Gamma_{ij}^k Z_i X_j \right) \frac{\partial}{\partial s_k} \\ &= \sum_{k=1}^{d} D_Z(X_k) \frac{\partial}{\partial q_k} + \sum_{k=1}^{d} D_Z(X_{k+d}) \frac{\partial}{\partial v_k} \\ &\quad + \frac{1}{2} \sum_{k,l=1}^{d} G^{v,kl} \left(\sum_{i,j,=1}^{d} (\partial_{q_j} G_{li}^v + \partial_{q_i} G_{lj}^v - \partial_{q_l} G_{ij}^v) Z_i X_j + (\partial_{v_i} G_{lj}^v) Z_{i+d} X_j \right) \frac{\partial}{\partial q_k} \end{aligned} \tag{11.32} $$

where $D_Z(\cdot)$ denotes the derivation with respect to Z. The above implies

$$ \begin{aligned} G({}^{G}\nabla_Z X, Y) &= \sum_{j,k=1}^{d} G_{ki}^v Y_k D_Z(X_i) + \sum_{j,k=1}^{d} G_{kj}^a Y_{k+d} D_Z(X_{j+d}) \\ &\quad + \frac{1}{2} \left(\sum_{i,j,k=1}^{d} (\partial_{q_j} G_{ki}^v + \partial_{q_i} G_{kj}^v - \partial_{q_k} G_{ij}^v) Z_i X_j Y_k + (\partial_{v_i} G_{kj}^v) Z_{i+d} X_j Y_k \right) \end{aligned} $$

Similarly, we can derive $G(X, {}^{G}\nabla_Z Y)$. Using the symmetry $G_{ij}^v = G_{ji}^v$, we can combine the previous results together and write

$G({}^{G}\nabla_Z X, Y) + G(X, {}^{G}\nabla_Z Y)$

[12]The section requirement on Z can be dropped.

$$
\begin{aligned}
&= \sum_{j,k=1}^{d} G^v_{ki} Y_k D_Z(X_i) + \sum_{j,k=1}^{d} G^a_{kj} Y_{k+d} D_Z(X_{j+d}) + \sum_{j,k=1}^{d} G^v_{ki} X_k D_Z(Y_i) + \sum_{j,k=1}^{d} G^a_{kj} X_{k+d} D_Z(Y_{j+d}) \\
&\quad + \frac{1}{2}\left(\sum_{i,j,k=1}^{d} (\partial_{q_j} G^v_{ki} + \partial_{q_i} G^v_{kj} - \partial_{q_k} G^v_{ij}) Z_i X_j Y_k + (\partial_{v_i} G^v_{kj}) Z_{i+d} X_j Y_k \right) \\
&\quad + \frac{1}{2}\left(\sum_{i,j,k=1}^{d} (\partial_{q_j} G^v_{ki} + \partial_{q_i} G^v_{kj} - \partial_{q_k} G^v_{ij}) Z_i Y_j X_k + (\partial_{v_i} G^v_{kj}) Z_{i+d} Y_j X_k \right) \\
&= \sum_{i,j=1}^{d} G^v_{ij} D_Z(X_i) Y_j + G^v_{ij} X_i D_Z(Y_j) + \sum_{i,j=1}^{d} G^a_{ij} D_Z(X_{d+i}) Y_{d+j} + G^a_{ij} X_{d+i} D_Z(Y_{d+j}) \\
&\quad + \sum_{i,j,k=1}^{d} X_i Y_j Z_k \partial_{q_k} G^v_{ij} + X_i Y_j Z_{k+d} \partial_{v_k} G^v_{ij} \\
&= \sum_{i,j=1}^{d} D_Z(G^v_{ij}) X_i Y_j + G^v_{ij} D_Z(X_i) Y_j + G^v_{ij} X_i D_Z(Y_j) + \sum_{i,j=1}^{d} G^a_{ij} D_Z(X_{d+i}) Y_{d+j} + G^a_{ij} X_{d+i} D_Z(Y_{d+j}) \\
&=^G \nabla_Z \left(\sum_{i,j=1}^{d} G^v_{ij} X_i Y_j + \sum_{i,j=1}^{d} G^a_{ij} X_{d+i} Y_{d+j} \right) =^G \nabla_Z G(X, Y)
\end{aligned}
$$

Therefore ${}^G\nabla$ is compatible with G.

So far we have proved the first statement of Theorem 11.5.3 that ${}^G\nabla$ is the unique solution that is compatible with G and satisfies the asymmetric condition. Below we show the expression of $\mathrm{pr}_3({}^G\nabla_{\ddot{q}}\ddot{q})$, where we recall $\ddot{q}(t)$ is a curve in $TT\mathcal{C}$. We use (11.32). By definition of pr_3 it extracts the parts on $\{\frac{\partial}{\partial q_i}\}_{i=1}^d$. Therefore, suppose we choose some chart on $\mathcal{C}$ of interest and we can write $\mathrm{pr}_3({}^G\nabla_{\ddot{q}}\ddot{q})$ as

$$
\begin{aligned}
&\mathrm{pr}_3({}^G\nabla_{\ddot{q}}\ddot{q}) \\
&= \sum_{k}^{d} \left(D_Z(X_k) + \sum_{l=1}^{d} \frac{1}{2} G^{v,kl} \sum_{i,j,=1}^{d} (\partial_{q_j} G^v_{li} + \partial_{q_i} G^v_{lj} - \partial_{q_l} G^v_{ij}) Z_i X_j + (\partial_{v_i} G^v_{lj}) Z_{i+d} X_j \right) \frac{\partial}{\partial q_k} \\
&= \sum_{k}^{d} \left(\ddot{q}_k + \sum_{l=1}^{d} \frac{1}{2} G^{v,kl} \sum_{i,j,=1}^{d} (\partial_{q_j} G^v_{li} + \partial_{q_i} G^v_{lj} - \partial_{q_l} G^v_{ij}) \dot{q}_i \dot{q}_j + (\partial_{v_i} G^v_{lj}) \ddot{q}_i \dot{q}_j \right) \frac{\partial}{\partial q_k} \\
&= \sum_{k}^{d} a_{\mathbf{G};k} \frac{\partial}{\partial q_k}
\end{aligned}
$$

where $a_{\mathbf{G};k}$ is the kth element of $\mathbf{a}_{\mathbf{G}} \coloneqq \ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}})$. ■

11.B.6 Proof of Theorem 11.5.4

Theorem 11.5.4. *Suppose $\mathcal{C}$ is related to K leaf-node task spaces by maps $\{\psi_i : \mathcal{C} \to \mathcal{T}_i\}_{i=1}^K$ and the ith task space $\mathcal{T}_i$ has an affine connection ${}^{G_i}\nabla$ on $T\mathcal{T}_i$, as defined in Theorem 11.5.3, and a covector function F_i defined by some potential and damping as described above. Let ${}^{G}\bar{\nabla} = \sum_{i=1}^K T\psi_i^{*G_i}\nabla$ be the pullback connection, $G = \sum_{i=1}^K T\psi_i^* G_i$ be the pullback metric, and $F = \sum_{i=1}^K T\psi_i^* F_i$ be the pullback covector, where $T\psi_i^* : T^*T\mathcal{T}_i \to T^*T\mathcal{C}$. Then ${}^{G}\bar{\nabla}$ is compatible with G, and $\mathrm{pr}_3({}^{G}\bar{\nabla}_{\dot{q}}\ddot{q}) = (\mathrm{pr}_3 \circ G^\sharp \circ F)(s)$ can be written as $\ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q},\dot{\mathbf{q}})^{-1}(\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q},\dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q},\dot{\mathbf{q}})\ddot{\mathbf{q}}) = -\mathbf{G}(\mathbf{q},\dot{\mathbf{q}})^{-1}(\nabla_{\mathbf{q}}\Phi(\mathbf{q}) + \mathbf{B}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}})$. In particular, if G is velocity-independent, then ${}^{G}\bar{\nabla} = {}^{G}\nabla$.*

Proof of Theorem 11.5.4. Let $\mathcal{T} = \mathcal{T}_1 \times \cdots \times \mathcal{T}_K$ and $\tilde{G}$ be the induced metric on $T\mathcal{T}$ by $\{G_i\}_{i=1}^K$. In addition, let $\psi : \mathcal{C} \to \mathcal{T}$ be the equivalent expression of $\{\psi_i\}$. Again we focus on the tangent bundle not the base manifold. Recall the definition of a pullback connection[13] $T\psi^{*\tilde{G}}\nabla$ is

$$T\psi_*(T\psi^{*\tilde{G}}\nabla_X Y) = \mathrm{pr}^{\tilde{G}}_{T\psi_*}\left({}^{\tilde{G}}\nabla_{T\psi_* X} T\psi_* Y\right) \tag{11.33}$$

for all sections X and Y on $TT\mathcal{C}$, where $\mathrm{pr}^{\tilde{G}}_{T\psi_*}$ is the projection onto the distribution spanned by $T\psi_*$ with respect to $\tilde{G}$, i.e. $\tilde{G}(T\psi_* X, \mathrm{pr}^{\tilde{G}}_{T\psi_*}(Z)) = \tilde{G}(T\psi_* X, Z)$ for all $X \in TT\mathcal{C}$ and $Z \in TT\mathcal{T}$. Note by the construction of the product manifold $\mathcal{T}$, $T\psi^{*\tilde{G}}\nabla = \sum_{i=1}^K T\psi_i^{*G_i}\nabla$.

Compatibility We show that $T\psi^{*\tilde{G}}\nabla$ is compatible with the pullback metric G. Let X, Y, Z be arbitrary sections on $TT\mathcal{C}$ and recall the definition of the pullback metric

$$G(X,Y) = T\psi^*\tilde{G}(X,Y) = \tilde{G}(T\psi_* X, T\psi_* Y)$$

[13] We note the distinction between $\psi^* : T^*\mathcal{T} \to T^*\mathcal{C}$ and $T\psi^* : T^*T\mathcal{T} \to T^*T\mathcal{C}$.

To show that $T\psi^{*\tilde{G}}\nabla$ is compatible, we derive an expression of $G(T\psi^{*\tilde{G}}\nabla_Z X, Y)$:

$$\begin{aligned} G(T\psi^{*\tilde{G}}\nabla_Z X, Y) &= T\psi^* \tilde{G}(T\psi^{*\tilde{G}}\nabla_Z X, Y) \\ &= \tilde{G}(T\psi_*(T\psi^{*\tilde{G}}\nabla_Z X), T\psi_* Y) \\ &= \tilde{G}\left(\mathrm{pr}^{\tilde{G}}_{T\psi_*}\left({}^{\tilde{G}}\nabla_{T\psi_* Z} T\psi_* X\right), T\psi_* Y\right) \\ &= \tilde{G}\left({}^{\tilde{G}}\nabla_{T\psi_* Z} T\psi_* X, T\psi_* Y\right) \end{aligned}$$

where we use (11.33) and the definition of projection. Using the above equation, we can see the compatibility easily:

$$\begin{aligned} &G(T\psi^{*\tilde{G}}\nabla_Z X, Y) + G(X, T\psi^{*\tilde{G}}\nabla_Z Y) \\ &= \tilde{G}\left({}^{\tilde{G}}\nabla_{T\psi_* Z} T\psi_* X, T\psi_* Y\right) + \tilde{G}\left(T\psi_* X, {}^{\tilde{G}}\nabla_{T\psi_* Z} T\psi_* Y\right) \\ &= {}^{\tilde{G}}\nabla_{T\psi_* Z}\tilde{G}(T\psi_* X, T\psi_* Y) \\ &= \psi^{*G}\nabla_Z \tilde{G}(T\psi_* X, T\psi_* Y) \\ &= \psi^{*G}\nabla_Z G(X, Y) \end{aligned}$$

Coordinate expression The coordinate expression of the pullback metric can be derived by its definition in (11.33), and the expression for the pullback covector is standard. For the pullback connection, similar to the proof of Theorem 11.5.3, we can show that $\mathrm{pr}_3({}^G\bar{\nabla}_{\ddot{q}}\ddot{q})$ can be written as $\ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}})$. In other words, the structured GDS equations are the coordinate expression of the pullback connection $T\psi^{*\tilde{G}}\nabla$, where the structure structure $\mathcal{S}$ is induced through the recursive application of `pullback` in RMPflow. Note that this is in general different from the connection of the pullback metric ${}^G\nabla$, which by Theorem 11.5.3 instead defines the unstructured GDS equation $\ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}})$.

Commutability However, in the special case when G is velocity-independent, we show that they are equivalent. That is, the pullback connection $T\psi^{*\tilde{G}}\nabla$ is equal to the connection

of the pullback matrix ${}^G\nabla$. This property is early shown in Theorem 11.5.1, which shows that in the velocity-independent case there is no need to distinguish structures. To prove this, we first note that ${}^G\nabla$ becomes symmetric as G is velocity-independent. As it is also compatible with G, we know that ${}^G\nabla$ is the Levi-Civita connection with respect to G. (Recall G is the Riemannian metric on the tangent bundle.) On the other hand, knowing that $T\psi^{*\tilde{G}}\nabla$ is compatible, to show that ${}^G\nabla = T\psi^{*\tilde{G}}\nabla$ we only need to check if $T\psi^{*\tilde{G}}\nabla$ is symmetric. Without further details, we note this is implied by the proof of Theorem 11.5.1. Therefore, we have $T\psi^{*\tilde{G}}\nabla =^G \nabla$. ■

11.C Degenerate GDSs

We discuss properties of degenerate GDSs. Let us recall the GDS $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$ means the differential equation

$$\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}}\Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} \tag{11.34}$$

where $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})$. For degenerate cases, $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ can be singular and (11.34) define rather a family of differential equations. Degenerate cases are not uncommon; for example, the leaf-node dynamics could have $\mathbf{G}$ being only positive semidefinite. Having degenerate GDSs does not change the properties that we have proved, but one must be careful about whether differential equation satisfying (11.34) exist. For example, the existence is handled by the assumption on $\mathbf{M}$ in Theorem 11.5.1 and the assumption on $\mathbf{M}_r$ in Corollary 11.5.1. For RMPflow, we only need that $\mathbf{M}_r$ at the root node is non-singular. In other words, the natural-form RMP created by `pullback` at the root node can be resolved in the canonical-form RMP for policy execution. A sufficient and yet practical condition is provided in Theorem 11.5.2.

To represent the above equation as an RMP, which is useful if we choose to design behavior by directly defining $\frac{d\widetilde{\Phi}}{ds}$, we can write it in the natural form as $[-w\frac{d\widetilde{\Phi}}{ds} - \xi, m]^{\mathbb{R}_+}$.

Note that the curvature term ξ behaves as a nonlinear damping term, slowing the system (from the perspective of the configuration space) as it approaches obstacles and vanishing when moving away from obstacles. Consequently, it biases the system toward curving along isocontours of the distance field. See Fig. 11.3 for a demonstration of these terms in isolation and in coordination with an obstacle repulsion potential.

CHAPTER 12

RMPFLOW WITH LEARNABLE LYAPUNOV FUNCTION RESHAPING

12.1 Introduction

Motion planning and control are core techniques to robotics (Correll et al., 2018; Johnson et al., 2015; Urmson et al., 2008). Ideally a good algorithm must be both computationally efficient and capable of navigating a robot safely and stably across a wide range of environments. Several systems were recently proposed to address this challenge (Cheng et al., 2018b; Kappler et al., 2018; Mukadam et al., 2017) through closely integrating planning and control techniques. In particular, RMPflow in Chapter 12 (Cheng et al., 2018b) is designed to combine reactive policies (Ijspeert et al., 2013; Khatib, 1987; Lo, Cheng, and Huang, 2016; Nakanishi et al., 2008; Peters et al., 2008) and planning (Ratliff, Toussaint, and Schaal, 2015b). Based on differential geometry, RMPflow offers a unified treatment of the nonlinear geometries arising from a robot's internal kinematics and task spaces (e.g. environments with obstacles). Given user-provided subtask motion policies expressed in the form of Riemannian Motion Policies (RMPs) (Ratliff, Issac, and Kappler, 2018) (i.e. a second-order motion policy along with a matrix function that acts as a directional importance weight), RMPflow can synthesize a global motion policy for the full task in an efficient and geometrically consistent manner and has desirable properties such as stability and being coordinate-free (Cheng et al., 2018b).

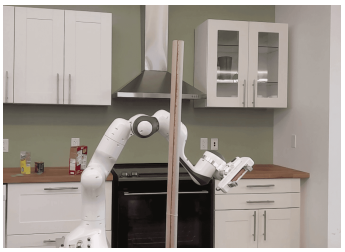
RMPflow has been successfully applied in many applications (Li et al., 2019a,b; Meng et al., 2019; Paxton et al., 2019; Sutanto et al., 2019). But RMPflow is not perfect. Despite its advancement, practical usage difficulties remain. For instance, the user must provide RMPs with matrix functions that properly describe the characteristics of subtask motion policies in order to build an effective RMPflow system. Otherwise, the final global policy

may have unsatisfactory performance, though still being geometrically consistent (with respect to some meaningless geometric structure). This poses a challenge for practitioners who are inexperienced in control systems, or for designing policies of tasks where the full state is hard to describe.

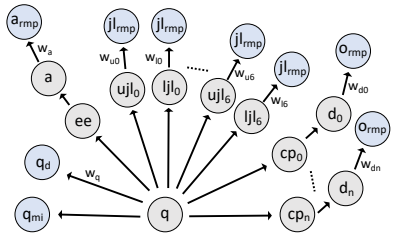
In this chapter we introduce a hierarchical Lyapunov function reshaping scheme into RMPflow to remedy the requirement of providing high-quality subtasks RMPs from the user. The modified algorithm, called RMPfusion, adds a set of multiplicative weight *functions* in the policy fusion step of RMPflow, which can be manually parametrized or modeled by function approximators (like neural networks). In a high level, these weight functions let RMPfusion adapt between multiple versions of RMPflow according to the robot's configuration and the environment. (RMPflow is RMPfusion with constant weights.) Therefore, an immediate benefit of our new algorithm is the extra design flexibility added to RMPflow. Compared with RMPflow, RMPfusion allows the user to start with simpler subtask RMPs and gradually build up more complex behaviors through the use of weight functions.

Interestingly, these weight functions in general do not just linearly combine outputs of motion policies as in (Arkin, 2008; Slotine, 1991). Instead they hierarchically reshape the inherent Lyapunov functions of the provided subtask policies, overall giving a nonlinear effect on the global policy RMPfusion creates. We prove that RMPfusion produces a policy that is Lyapunov-stable with respect to this reshaped Lyapunov function given by the weight functions. Therefore, the overall the system is stable, as long as the weight functions are non-negative and non-degenerate.

These properties suggest that we can treat RMPfusion as a structured policy class in reinforcement/imitation learning and optimize the weight functions to improve the combined policy's performance. Importantly, as RMPfusion remains stable under minor restriction on weight functions, we arrive at a policy class that is guaranteed to be stable, even during the immature phase of learning. Thus, RMPfusion is suitable for learning with safety



(a) A snapshot of the experiment.



(b) The RMP-tree* used for the Franka robot.

Figure 12.1: Franka robot navigating around an obstacle using RMPfusion with the RMP-tree*. Gray nodes show task spaces, blue nodes show subtask RMPs, and weight functions are shown on the respective edges where they are defined. See Section 12.4.2 for details.

constraints; for example, we can ensure that certain safe policies (like collision avoidance) are the only ones activated when the robot is facing extreme conditions. These theoretical properties of RMPfusion are verified in imitation learning tasks, in both simulations and on a real-world robot (Figure 12.1). Not only did RMPfusion learn to mimic the expert policy, but it also yielded stable policies throughout the learning. This chapter is partly based on our paper published as (Mukadam et al., 2019).

12.2 Quick Recap of RMPflow

12.2.1 Computation

Inspired by geometric control theory (Bullo and Lewis, 2004), RMPflow provides a rigorous framework for policy fusion with theoretical guarantees, such as stability and geometric consistency. In implementation, RMPflow is realized by a data structure called *RMP-tree*, and a set operations called *RMP-algebra*. Below we highlight major features of each component.

RMP-tree An RMP-tree (e.g. Fig. 12.1b but without the weights $w_.$) is a directed tree, which expresses the task map ψ as a sequence of basic maps. The RMP-tree serves two major purposes: (i) it provides a language for the user to specify the connections between different subtasks, and (ii) it allows RMPflow to reuse those basic computations inside ψ to achieve efficient policy fusion. In the RMP-tree, each node represents an RMP and

its state; and each edge represents a transformation between manifolds in the user given decomposition of ψ. Particularly, the leaf nodes consist of the user-defined subtask RMPs, and the root node maintains the RMP of the global policy π on $\mathcal{C}$.

RMP-tree uses RMP (Ratliff, Issac, and Kappler, 2018) to describe motion policies on manifolds. Consider an m-dimensional manifold $\mathcal{M}$ with coordinate $\mathbf{x} \in \mathbb{R}^m$ and a motion policy $\mathbf{a}$ on $\mathcal{M}$ (i.e. $\ddot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, \dot{\mathbf{x}})$). An RMP pairs the motion policy $\mathbf{a}$ with an *abstract* inertia matrix $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}_+^{m \times m}$, a function of *both* $\mathbf{x}$ and $\dot{\mathbf{x}}$ that describes the directional importance of $\mathbf{a}$ at the current state $(\mathbf{x}, \dot{\mathbf{x}})$ (see Chapter 11 for details). The RMP of $\mathbf{a}$ can be written in the canonical form $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ or in the natural form $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$, in which $\mathbf{f} = \mathbf{M}\mathbf{a}$ is called the force map. Note that $\mathbf{f}$ and $\mathbf{M}$ are not necessarily physical quantities, and that the motion policy in an RMP is not necessarily in the form of a mass-spring-damper system.

RMP-algebra RMPflow uses the RMP-algebra to combine the subtask policies at leaf nodes into a global policy on the configuration space at the root node. RMP-algebra consists of three operators:

(i) `pushforward` propagates the state $(\mathbf{x}, \dot{\mathbf{x}})$ of a node in the RMP-tree to update the states of its K child nodes. The state of its ith child node is computed as $(\mathbf{y}_i, \dot{\mathbf{y}}_i) = (\psi_i(\mathbf{x}), \mathbf{J}_i(\mathbf{x})\dot{\mathbf{x}})$, where ψ_i is the transformation $\mathbf{y}_i = \psi_i(\mathbf{x})$ and $\mathbf{J}_i = \partial_{\mathbf{x}}\psi_i$ is the Jacobian matrix.

(ii) `pullback` propagates the RMPs from the K child nodes to the parent node as $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ with

$$\mathbf{f} = \sum_{i=1}^{K} \mathbf{J}_i^\top (\mathbf{f}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{x}}) \qquad \text{and} \qquad \mathbf{M} = \sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i \tag{12.1}$$

where $[\mathbf{f}_i, \mathbf{M}_i]^{\mathcal{N}_i}$ is the RMP of the ith child node in the natural form.

(iii) `resolve` maps an RMP from its natural form $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ to its canonical form $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ with $\mathbf{a} = \mathbf{M}^\dagger \mathbf{f}$, where $\dagger$ denotes Moore-Penrose inverse.

To compute the global policy π at time t, RMPflow first performs a forward pass by

recursively calling `pushforward`. Then it performs a backward pass by recursively calling `pullback` and computes $[\mathbf{f}_{\mathrm{r}}, \mathbf{M}_{\mathrm{r}}]^{\mathcal{C}}$ at the root. Finally, the global policy $\pi = \mathbf{a}_{\mathrm{r}}$ is generated by using `resolve`. Loosely speaking, the global policy π can be viewed as a weighted combination of the subtask policies. This can be seen by rewriting (12.1) as $\mathbf{a} = \mathbf{M}\mathbf{f} = (\sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i)^{-1} \mathbf{J}_i^\top (\mathbf{M}_i \mathbf{a}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{x}})$ (which is linear combination of child policies $\mathbf{a}_i$ plus some curvature correction due to $\dot{\mathbf{J}}_i$).

12.2.2 Theoretical Properties of RMPflow and GDSs

In Chapter 11, RMPflow is proved to be Lyapunov stable and coordinate-free, when the subtask policies belong to Geometric Dynamical Systems (GDSs) (Cheng et al., 2018b). GDSs are a family of dynamical systems on manifolds that generalizes Simple Mechanical Systems to have *velocity-dependent* inertias. Let $\mathcal{M}$ be an m-dimensional manifold with coordinate $\mathbf{x} \in \mathbb{R}^m$. Let $\mathbf{G} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m \times m}$ be a *metric* matrix, $\mathbf{B} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m \times m}$ be a *damping* matrix, and $\Phi : \mathbb{R}^m \to \mathbb{R}$ be a *potential* function, which is lower bounded. A dynamical system on $\mathcal{M}$ is said to be a *GDS* $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$ if it follows

$$\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}, \tag{12.2}$$

in which $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})$, $\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \frac{1}{2} \sum_{i=1}^{m} \dot{x}_i \partial_{\dot{\mathbf{x}}} \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})$, $\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \overset{\mathbf{x}}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} - \frac{1}{2} \nabla_{\mathbf{x}} (\dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}})$, and $\overset{\mathbf{x}}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq [\partial_{\mathbf{x}} \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}]_{i=1}^{m}$. The term $\mathbf{M}$ is again called the inertia matrix, despite being a function of both $\mathbf{x}$ and $\dot{\mathbf{x}}$. The *curvature* terms $\boldsymbol{\Xi}(\mathbf{x}, \dot{\mathbf{x}})$ and $\boldsymbol{\xi}(\mathbf{x}, \dot{\mathbf{x}})$ are generated from the dependency of $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ on $\mathbf{x}$ and $\dot{\mathbf{x}}$; if $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$, then $\mathbf{G}(\mathbf{x}) = \mathbf{M}(\mathbf{x})$ and $\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$ in (11.22). In view of this, a GDS extends (11.22) to have general potentials and velocity-dependent metrics, which is useful in modeling collision avoidance behaviors (Cheng et al., 2018b).

The behavior of a GDS $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$ is characterized by the Lyapunov function

$$V(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{2} \dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \Phi(\mathbf{x}). \tag{12.3}$$

In Chapter 11, it is shown that the stability property of RMPflow is governed by a Lyapunov function in a similar form (Cheng et al., 2018b), when the leaf-node policies are GDSs. An RMP $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ is a GDS if its motion policy is $\mathbf{a} = \mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})^{-1}(-\nabla_{\mathbf{x}}\Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} - \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}))$.

Theorem 12.2.1. (Cheng et al., 2018b) *Suppose an RMP-tree has K leaf nodes of GDSs $(\mathcal{T}_k, \mathbf{G}_k, \mathbf{B}_k, \Phi_k)$ with Lyapunov function V_k in* (12.3)*, for $k = 1, \ldots, K$. Let $V_r = \sum_{k=1}^{K} V_k$ be a Lyapunov candidate.*

1. *If $\mathbf{M}_r$ of the root-node RMP on $\mathcal{C}$ is positive definite, then $\dot{V}_r = -\sum_{k=1}^{K} \dot{\mathbf{x}}_k^{\top}\mathbf{B}_k\dot{\mathbf{x}}_k \leq 0$.*

2. *If further $\sum_{k=1}^{K} \mathbf{J}_k^{\top}\mathbf{G}_k\mathbf{J}_k \succ 0$ and $\sum_{k=1}^{K} \mathbf{J}_k^{\top}\mathbf{B}_k\mathbf{J}_k \succ 0$, the system converges forwardly to $\{(\mathbf{q}, \dot{\mathbf{q}}) : \nabla_{\mathbf{q}}\Phi_r(\mathbf{q}) = 0, \dot{\mathbf{q}} = 0\}$, where $\mathbf{J}_k = \partial_{\mathbf{q}}\mathbf{x}_k$ and $\Phi_r(\mathbf{q}) = \sum_{k=1}^{K} \Phi_k(\mathbf{x}_k(\mathbf{q}))$.*

12.3 RMPfusion

RMPflow provides a control-theoretic framework for combining subtask policies. However, certain limitations exist. Particularly, it requires the user to provide sensible inertia matrices (cf. Chapter 11) to describe the subtask policies' characteristics in the leaf-nodes RMPs; failing to do so may result in a global policy with undesirable performance, albeit still being geometrically consistent with the meaningless geometric structure induced by the bad inertia matrices.

In this chapter, we propose a modified algorithm, RMPfusion, which adds extra flexibilities into RMPflow to address this difficulty. The main idea is to introduce an additional set of weight functions as gates to switch on and off the child-node policies in the RMP-tree, based on the current state of the robot and the environment. These functions can either be designed by hand, or be parameterized as function approximators (like neural networks) which are then learned end-to-end from data (see Section 12.3.4). As a result, RMPfu-

sion can combine simpler/imperfect subtask RMPs into a better global policy, lessening the burden on the user to directly provide high-quality subtasks RMPs.

RMPfusion modifies RMP-tree and RMP-algebra into RMP-tree* and RMP-algebra*, respectively. RMP-tree* augments each node in RMP-tree with extra information and each edge with a weight function; RMP-algebra* replaces `pullback` with `pullback`*. Below we define these modifications. In addition, we show that RMPfusion retains the nice structural properties of RMPflow: under mild conditions on the weight functions, the global policy of RMPfusion is Lyapunov stable. Later in Section 12.3.4, we will show how to learn the weight functions in RMPfusion from data.

12.3.1 RMP-tree* and RMP-algebra*

Modified node In addition to the RMP and its state, each node in RMP-tree* also stores the *values* of a scalar function L and the metric matrix $\mathbf{G}$. When a leaf-node RMP is a GDS, $\mathbf{G}$ is defined as (13.3) and $L = \frac{1}{2}\dot{\mathbf{x}}^\top \mathbf{G}\dot{\mathbf{x}} - \Phi(\mathbf{x})$ (analogue of the Lagrangian in mechanical systems).

Modified edge Each edge in an RMP-tree* has in addition a weight function. This weight is a function of the parent-node configuration and some auxiliary state (which describes the task at hand, e.g., the location of the goal in a reaching task).

Modified pullback We modify `pullback` into `pullback`* to use the weight functions on edges to combine child-node RMPs. For the parent and child nodes given in (12.1), we set instead

$$\begin{aligned} \mathbf{f} &= \sum_{i=1}^{K} w_i \mathbf{J}_i^\top (\mathbf{f}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{x}}) + \mathbf{h}_i, \quad \mathbf{M} = \sum_{i=1}^{K} w_i \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i, \\ \mathbf{G} &= \sum_{i=1}^{K} w_i \mathbf{J}_i^\top \mathbf{G}_i \mathbf{J}_i, \quad L = \sum_{i=1}^{K} w_i L_i \end{aligned} \tag{12.4}$$

where $\mathbf{h}_i = L_i \nabla_{\mathbf{x}} w_i - (\dot{\mathbf{x}}^\top \nabla_{\mathbf{x}} w_i) \mathbf{J}_i^\top \mathbf{G}_i \mathbf{J}_i \dot{\mathbf{x}}$. From (12.4), we see that `pullback`* does *not* simply linearly combine child-node motion policies. It adds a correction term $\mathbf{h}_i$, which

is designed to anticipates the change of weighting w_i so that the system remains stable. When applied recursively in policy generation, it would *hierarchically* reshape the Lyapunov functions (see Section 12.3.3).

12.3.2 Stability

We show RMPfusion is also Lyapunov stable like RMPflow. To state the stability property, let us introduce additional notation to describe the functions in the RMP-tree*. We will use $(i; j)$ to denote the ith node in depth j of an RMP-tree* and we use $C_{(i;j)}$ to denote the indices of its child nodes. For example, node $(1; 0)$ denotes the root node (also denoted as $\mathtt{r}$). In addition, we will refer to the functions on the edges using the indices of the child nodes, e.g., the Jacobian of the transformation to the ith node in depth j is denoted as $\mathbf{J}_{(i;j)}$. We show the stability property of RMPfusion when all the leaf nodes are of GDSs, like Theorem 12.2.1. The proof is given in Section 12.A.

Theorem 12.3.1. *Suppose an RMP-tree* has leaf-node policies as GDSs with Lyapunov functions given as* (12.3)*. Define $V_{(i;j)}$, $\mathbf{B}_{(i;j)}$, and $\Phi_{(i;j)}$ on the RMP-tree* through the recursion*

$$V_{(i;j)} = \sum_{k \in C_{(i;j)}} w_{(k;j+1)} V_{(k;j+1)}, \quad \mathbf{B}_{(i;j)} = \sum_{k \in C_{(i;j)}} w_{(k;j+1)} \mathbf{J}_{(k;j+1)}^{\top} \mathbf{B}_{(k;j+1)} \mathbf{J}_{(k;j+1)}$$
$$\Phi_{(i;j)} = \sum_{k \in C_{(i;j)}} w_{(k;j+1)} \Phi_{(k;j+1)} \tag{12.5}$$

in which the boundary condition is given by the leaf-node GDSs. Let $V_{\mathtt{r}}$ be a Lyapunov candidate.

1. *If $\mathbf{M}_{\mathtt{r}} \succ 0$, then $\dot{V}_{\mathtt{r}} = -\dot{\mathbf{q}}^{\top} \mathbf{B}_{\mathtt{r}} \dot{\mathbf{q}} \leq 0$.*
2. *If further $\mathbf{G}_{\mathtt{r}}, \mathbf{B}_{\mathtt{r}} \succ 0$, the system converges forwardly to $\{(\mathbf{q}, \dot{\mathbf{q}}) : \nabla_{\mathbf{q}} \Phi_{\mathtt{r}}(\mathbf{q}) = 0, \dot{\mathbf{q}} = 0\}$.*

Theorem 12.3.1 shows that the system is Lyapunov stable with respect to $V_{\mathtt{r}}$. To satisfy the conditions required in Theorem 12.3.1, a sufficient condition is to select leaf-node GDSs with certain monotone metrics (Cheng et al., 2018b, Theorem 2) and have positive weight functions on edges. Therefore, in addition to the conditions needed by RMPflow, RMPfusion only imposes mild constraints on the weight functions. This is a useful feature when the weight functions are learned from data, because Theorem 12.3.1 essentially guarantees the output policy is always stable even in the premature stage of learning.

Note that it is straightforward to extend RMP-tree* to include, in (12.1), an extra time-varying term that vanishes as $t \to \infty$ (like the one used in DMPs (Ijspeert et al., 2013)) and to consider time-varying potentials (e.g. in tracking applications). We omit discussions about these generalizations due to space limitation.

12.3.3 Advantages of RMPfusion over RMPflow

RMPfusion strictly generalizes RMPflow. When each weight is constant one, RMPfusion becomes RMPflow (i.e. `pullback`* is the same as `pullback` and Theorem 12.3.1 reduces to Theorem 12.2.1). More generally, RMPfusion allows mixing policies through reweighting their Lyapunov functions, while retaining the nice structural properties of RMPflow, as shown in Theorem 12.3.1.

In comparison, RMPfusion has a more flexible way to express policies and compose the subtask Lyapunov functions into the Lyapunov candidate $V_{\mathtt{r}}$ in (12.6). Whereas Theorem 12.2.1 uses the simple summation of subtask energies $V_{\mathtt{r}} = \sum_{i=1}^{K} V_i$, Theorem 12.3.1 effectively uses the Lyapunov function

$$V_{\mathtt{r}} = \sum_{k_1 \in C_{(1;0)}} w_{(k_1;1)} \sum_{k_2 \in C_{(k_1;1)}} \cdots \sum_{k_D \in C_{(k_{D-1};D-1)}} w_{(k_D;D)} V_{(k_D;D)} \tag{12.6}$$

for a depth-D RMP-tree* (cf. (12.5)) and each weight $w_{(i;j)}$ can be a function of the configuration and auxiliary state of the parent of node $(i; j)$. Therefore, from (12.5) and (12.6),

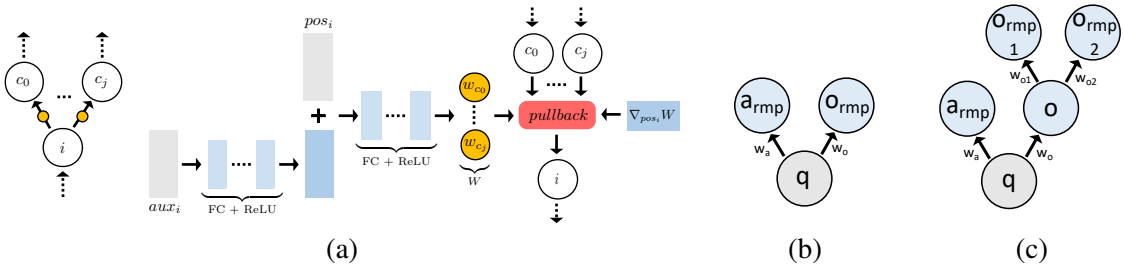


Figure 12.2: (a) Shows the network used for learning with RMPfusion, specifically for any node i on the RMP-tree*, with children $c_0, \ldots, c_j$. If i is a leaf node, then it is evaluated from the designed RMP policy. The global policy is obtained by applying `resolve` on the root node RMP. RMP-tree* used in experiments for (b) `2d1level` and (c) `2d2level`.

RMPfusion can be viewed as a form of hierarchical Lyapunov function reshaping scheme along the hierarchy structure induced by the RMP-tree*. Consequently, the recursive formulation of RMPfusion allows the user only to provide basic subtask policies and gradually increase their expressiveness by the weight functions. In contrast, using RMPflow requires directly specifying subtask policies with complicated behaviors. We include a concrete example to illustrate the benefit of this extra flexibility in Section 12.B.

12.3.4 Learning RMPfusion

We presented a new computational graph, RMPfusion, which supplements RMPflow with a set of multiplicative weight functions to achieve extra flexibility in policy fusion. In Section 12.C, we show these weight functions can be learned by back-propagation, and therefore RMPfusion can be treated as a parameterized policy class in policy optimization by using computational graph libraries like tensorflow (Abadi et al., 2016) or pytorch (Paszke et al., 2017). Finally, it is important to note that we do not have to learn all the weight functions in a RMP-tree*. If we know that certain leaf-node RMPs have to be turned on, we can adopt a semi-parametric scheme of weight functions. For example, we can design parameterization of the weight functions such that only collision avoidance RMPs are turned on, when the robot is extremely close to an obstacle. This property is due to the structure of RMP-tree*, which is interpretable, unlike policies purely based on general function approximators. Interpretability allows for prior knowledge (like constraints and preferences) to be easily incorporated into the policy structure. This feature is particularly valuable for

policy learning with safety constraints (Garcıa and Fernández, 2015).

12.4 Experiments

We validate our approach with experiments of imitation learning. The goal is to show that RMPfusion with an RMP-tree* that is parametrized by randomly initialized neural networks (as in Figure 12.2) is able to mimic the expert policy's behavior by observing expert demonstrations. This setup simulates the situation where the user of RMPfusion only provides imperfect subtask policies. We also use these experiments to validate the stability properties of RMPfusion by studying if the Lyapunov function of the policies generated by RMPfusion (even the premature ones obtained before learning converges) decay monotonically over time. We perform these experiments with a 2D particle robot and with a Franka Panda 7-DOF robot (video of experiments is available at https://youtu.be/McSrpW-mIq4).

As our aim it not to invent a new imitation learning algorithm, we adopt the most basic approach, behavior cloning (Pomerleau, 1989), in which the demonstrations are purely generated by running the expert policy alone without any active intervention from the learner. The objective of these experiments is to study how well RMPfusion can recover the behaviors of an expert that is within its effective policy class, and therefore we use a known RMP-tree* with fixed weights as the expert policy. We choose this setting to rule out bias due to mismatches between policy classes, because properly handling policy class biases in imitation learning is a non-trivial research question on its own right (Cheng and Boots, 2018; Cheng et al., 2018a; Ross and Bagnell, 2014; Ross, Gordon, and Bagnell, 2011). Note that any policy optimization technique can be used to train RMPfusion, including online imitation learning and policy gradient methods, etc.

12.4.1 2D Robot

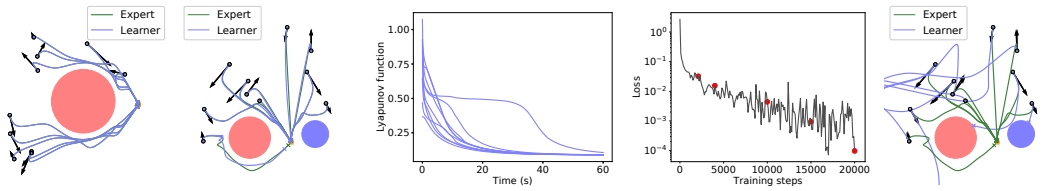
We first validate our approach on two problems where a 2D robot is tasked with reaching a goal while avoiding one obstacle (`2d1level`) or two obstacles (`2d2level`). The RMP-

tree* for these problems are shown in Figure 12.2b-12.2c and are detailed in Section 12.D. The tree structure here is heuristically chosen based on the problem domain, as in RMPflow and typically follows the robots kinematic chain and then extends into the workspace and abstract task spaces. In the `2d1level` problem, the aim is to show near-perfect recovery of the weights given that the problem is convex in the weight functions. The `2d2level` problem adds extra complexity to the learning process. It introduces multiplication between weights so the weights cannot be uniquely identified. The aim here is to show that close-to-expert behavior can still be achieved.

Data For each problem, the expert policy is generated by the respective RMP-tree* with some fixed assigned weights, which are unknown to the learner. The training data consist of 20 *randomly selected environments* with varying placements and sizes of obstacles. In each environment, the expert is run to generate 50 trajectories from unique initial states, and 60 temporally equidistant data points on each trajectory are recorded. Each data point is a pair of input and output: the input consists of the state (position and velocity) of the 2D particle and the auxiliary state (obstacle location and dimension, goal location) i.e. the meta information about the environment; the output consists of the action (acceleration) as specified by the expert given the input state visited by running the expert policy. Test data are collected by repeating this process with 5 new environments with 10 trajectories in each environment.

Unstructured network For `2d2level` we also compare our RMPfusion `learner-rmp` with an unstructured neural network `learner-un`. This is a fully connected feed forward network with similar number of learnable parameters compared to `learner-rmp`. This network takes robot state and auxiliary state as the inputs, and outputs the acceleration. Our aim with this comparison is to show that an unstructured approach cannot offer any stability or safety guarantees, and with the same amount of data and training underperforms compared to the structured approach.

Training We use the mean squared error between the action generated by any learner



(a) `2d1level` (b) `2d2level` (c) (d) (e) `2d2level`

Figure 12.3: Trajectories generated in by (a)-(b) `learner-rmp` and (e) `learner-un`, compared to the expert are shown. Initial state is a black circle for position and black arrow for velocity. The environment has obstacles (red and blue) and goal (orange square). (c) shows the corresponding Lyapunov function for `learner-rmp` trajectories in (b) while (d) shows its learning curve.

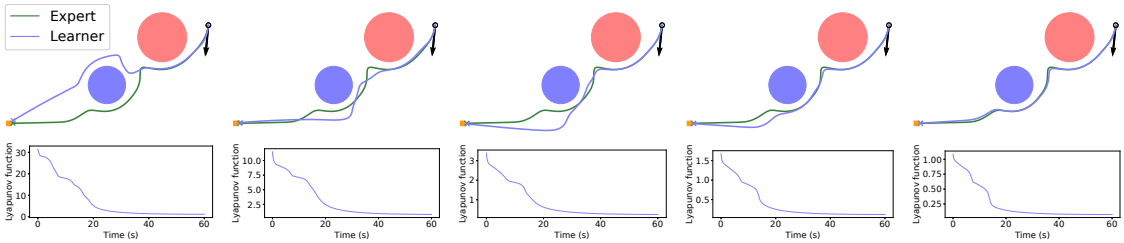


Figure 12.4: Improvement of the behavior produced by `learner-rmp` at various stages during training for `2d2level`. The top row shows the trajectories and the bottom row shows the corresponding Lyapunov function. From left to right these plots correspond to the red dots from left to right on the training curve in Figure 12.3d.

and the action specified by the expert as the loss function for imitation learning. All learners are trained using RMSprop (Tieleman and Hinton, 2012) with a minibatch size of 200 for 20,000 iterations. The number of iterations were chosen such that learning roughly converged and over-fitting had not happened.

Results We report two types of test loss: the batch-loss is the average loss on the entire test dataset generated by the expert policy, and the online-loss is the average loss at every time step (1 second interval) on the trajectories generated by the learner's policy starting from the initial states in the test dataset. In `2d1level`, the batch-loss is 5.42×10^{-5} and the online-loss is 5.82×10^{-5}. In `2d1level`, for `learner-rmp` the batch-loss is 2.45×10^{-4} and the online-loss is 2.78×10^{-4}, while for `learner-un` the batch-loss is 0.111 and the online-loss is 12.203. The higher batch-loss for `learner-un` indicates that with the same amount of data and training the network is unable to learn the policy from the expert, while the much worse online-loss indicates that it cannot generalize well

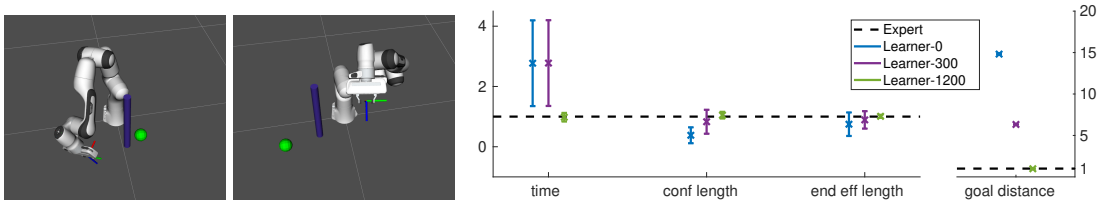
and succumbs to covariate shift problems.

Figures 12.3a, 12.3b and 12.3e show the evaluation of the trained networks on an example test environment. These results show that RMPfusion can perfectly match the behavior of the expert in the convex case (`2d1level`), while achieving near-expert performance in the non-identifiable case (`2d12evel`). From the overall results we also observe that `learner-un` is never able to reach the goal and also has a collision rate of 28% (e.g. Figure 12.3e), whereas `learner-rmp` successfully finishes the task 100% of the time. We also tried a unstructured network with 5.8 times the number of learnable parameters. While the loss values improved with a small drop in collision rate, it was still never able to complete the task (please see Section 12.D for more details). Figure 12.4 shows the improving progression of `learner-rmp` during training, in which each snapshot corresponds to an associated point on the training curve in Figure 12.3d. This verifies that with training we can progressively improve the behavior of the learner. In addition, we verify that the stability properties of RMPfusion in the associated Lyapunov functions in Figure 12.3c and Figure 12.4. We see that, regardless of the setting, the Lyapunov functions always decays monotonically as indicated by Theorem 12.3.1. This suggests RMPfusion produces a stable policy even when the learned weight functions are premature before the learning has converged (Figure 12.4). On the other hand, `learner-un` does not always avoid collision or provide any stability during or after training (see Figure 12.7 in Section 12.D).

12.4.2 Franka Robot

We also validate our approach in a more realistic setup with a Franka Panda 7-DOF robot arm. In these experiments, the task is to reach a goal while navigating around an obstacle. The RMP-tree* used is shown in Figure 12.1b, where the configuration space of the robot is the root node, and weights functions are shown on the edges where they are defined. Please see Section 12.D for details.

Data and training The expert policy is given by the RMP-tree* with some fixed but unknown weights, while the learner's policy is defined by the RMP-tree* with neural net-



(a) (b)

Figure 12.5: (a) An example from the training dataset (left) and the test dataset (right). The robot is shown in its start configuration with an obstacle (cylinder) and a goal (sphere). (b) Learner's performance with respect to the expert on the test dataset for the experiments with the Franka robot.

work weight functions that will be learned through behavior cloning. For training data we place an obstacle in a fixed location near the robot and sample different start configurations and goal locations that are in a region in front of the obstacle from the robot's perspective, so that the robot is forced to interact with the obstacle while trying to reach the goal. We run the expert to generate 110 unique trajectories for the training data. The trajectories are 5-10 seconds long and data is collected every 0.1 seconds; a data point consists of the state (configuration position and velocity of the robot), the auxiliary state (distances to goal and obstacle), and the expert action (acceleration). In a new environment with a different placement of the obstacle, this process is repeated to gather the test dataset where the expert is used to generate 20 unique trajectories. An example from the training and test dataset is shown in Figure 12.5a. The loss function is the same as in the experiments with the 2D robot and we train the policy using ADAM (Kingma and Ba, 2014) with a minibatch size of 200 for 1500 iterations. The number of iterations were chosen such that learning roughly converged and over-fitting had not happened.

Results We compare the performance of the learner, against the expert, at various stages of training: `learner-0` at no training (the neural network is initialized with random weights), `learner-300` at 300 iterations, and `learner-1200` at 1200 iterations when the learning converges. We record the following metrics on the test dataset for the expert and all the learners: (i) time: the time to reach within a precision of $0.05m$ of the goal; we time-out the execution at 10 seconds, (ii) conf length: the distance traveled in configuration space, (iii) end eff length: the distance traveled by the end effector in workspace, and

(iv) goal distance: the distance to the goal from the end effector at the end of an execution.

Figure 12.5b shows the performance of the learners relative to the expert on the test dataset (it plots the mean and the standard deviation of the ratios of the learner's metric and the expert's metric across trajectories; the expert is shown as the dotted horizontal baseline). From these results we see that, when the learner is not trained, the robot does not move much and incurs a high goal distance before timing out. With more training, the goal error reduces as the robot start traveling towards the goal but it still often times out. As the learning converges so does the performance of the learner towards the expert's performance. In all the trajectories across all the learners there are no collision, which verifies that constraints like safety can be incorporated through the structured learning approach that RMPfusion allows. We do a qualitative comparison on an example execution with the expert and the learners and also verify the stability properties of RMPfusion (even during learning) with the monotonically decreasing Lyapunov functions on these executions. Please see Section 12.D and Figure 12.9 therein for details.

12.5 Conclusion

We introduce extra parametrization flexibility into RMPflow and propose a new algorithm called RMPfusion. RMPfusion features a set of learnable weight functions that specifies the importance of subtask policies based on the robot's configuration and the environment. Consequently, RMPfusion can combine imperfect subtask policies into a global policy with good performance, where the original RMPflow fails. We demonstrate the ability of RMPfusion to learn weight functions for policy fusion in experiments, and further theoretically prove that RMPfusion inherits the Lyapunov-type stability from RMPflow with only mild conditions on the weight functions. These structural properties and encouraging experimental results of RMPfusion suggest that RMPfusion can be treated as a class of structural policies suitable for policy learning with safety and interpretability requirements. Important future work includes designing more expressive policies based on RMPfusion, e.g., we

can modify RMPfusion slightly to also learn part of the subtask policies and extra perturbations.

12.A Proof of Theorem 12.3.1

We provide the proof of Theorem 12.3.1 for completeness. We use (Cheng et al., 2018b, Thoerem 1) as the main lemma in our proof.

12.A.1 Background

We first recall the definition of structured GDS (Cheng et al., 2018b), which augments a GDS with the information on how the metric matrix $\mathbf{G}$ factorizes, in order to state (Cheng et al., 2018b, Thoerem 1).

Definition 12.A.1. (Cheng et al., 2018b) Suppose $\mathbf{G}$ has a structure $\mathcal{S}$ that factorizes $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{J}(\mathbf{x})^\top \mathbf{H}(\mathbf{z}, \dot{\mathbf{z}}) \mathbf{J}(\mathbf{x})$, where $\mathbf{z} : \mathbf{x} \mapsto \mathbf{z}(\mathbf{x}) \in \mathbb{R}^n$ and $\mathbf{H} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+^{n \times n}$, and $\mathbf{J}(\mathbf{x}) = \partial_{\mathbf{x}} \mathbf{z}$. The tuple $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ is a *structured GDS*, if

$$\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}} + \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} \tag{12.7}$$

where $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \mathbf{J}(\mathbf{x})^\top (\boldsymbol{\xi}_{\mathbf{H}}(\mathbf{z}, \dot{\mathbf{z}}) + (\mathbf{H}(\mathbf{z}, \dot{\mathbf{z}}) + \boldsymbol{\Xi}_{\mathbf{H}}(\mathbf{z}, \dot{\mathbf{z}}))\dot{\mathbf{J}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}})$. Given two structures, $\mathcal{S}_a$ is said to *preserve* $\mathcal{S}_b$ if $\mathcal{S}_a$ has the factorization (of $\mathbf{H}$) made by $\mathcal{S}_b$.

As noted in (Cheng et al., 2018b), GDSs are structured GDSs with a trivial structure (i.e. $\mathbf{z} = \mathbf{x}$), and structured GDSs reduce to GDSs if $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$, or if the manifold is one-dimensional.

Lemma 12.A.1. *(Cheng et al., 2018b, Thoerem 1) Suppose the ith child node follows* $(\mathcal{N}_i, \mathbf{G}_i, \mathbf{B}_i, \Phi_i)_{\mathcal{S}_i}$ *and has coordinate* $\mathbf{y}_i$*. Let* $\mathbf{a}_i = (\mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i})^\dagger (-\boldsymbol{\eta}_{\mathbf{G}_i;\mathcal{S}_i} - \nabla_{\mathbf{y}_i} \Phi_i - \mathbf{B}_i \dot{\mathbf{y}}_i)$ *and* $\mathbf{M}_i = \mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i}$*. Suppose* $\mathbf{a}$ *of the parent node is given by* `pullback` *with* $\{(\mathbf{a}_i, \mathbf{M}_i)_C^{\mathcal{N}_i}\}_{i=1}^K$*. Then* $\mathbf{a}$ *follows the* pullback structured GDS $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$*, where* $\mathbf{G} =$

$\sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{G}_i \mathbf{J}_i$, $\mathbf{B} = \sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{B}_i \mathbf{J}_i$, $\Phi = \sum_{i=1}^{K} \Phi_i \circ \mathbf{y}_k$, *$\mathcal{S}$ preserves $\mathcal{S}_i$, and* $\mathbf{J}_i = \partial_{\mathbf{x}} \mathbf{y}_i$. *That is, the parent node is* $(\mathbf{a}, \mathbf{M})_C^{\mathcal{M}}$ *such that* $\mathbf{M} = \sum_{i=1}^{K} \mathbf{J}_i^\top (\mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i}) \mathbf{J}_i$ *and* $\mathbf{a} = (\mathbf{G} + \boldsymbol{\Xi}_{\mathbf{G}})^\dagger (-\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} - \nabla_{\mathbf{x}} \Phi - \mathbf{B}\dot{\mathbf{x}})$.

Lemma 12.A.1 shows that the original `pullback` operator preserves structured GDSs. Consequently, when all the leaf nodes are GDSs, the root node is a structured GDS, which implies the type of Lyapunov stability in Theorem 12.2.1.

12.A.2 Proof of Theorem 12.3.1

We prove the stability of RMPfusion using similar techniques as (Cheng et al., 2018b). Using the recursive property, it is sufficient to show that `pullback`* preserves a family of structured GDSs, which are specified by the weight functions. Then the statement of Theorem 12.3.1 follows directly as in (Cheng et al., 2018b).

We proceed by first decoupling the `pullback`* into two steps. Let u be a parent node on manifold $\mathcal{M}$ and $\{v_k\}_{k=1}^K$ be its K child nodes on manifold $\{\mathcal{N}_k\}_{k=1}^K$ in an RMP-tree*. Between u and each v_k, we add an extra node $\tilde{v}_k$ on manifold $\mathcal{M}$ to create a new graph. In this new graph, u has K child nodes $\{\tilde{v}_k\}_{k=1}^K$ *with identity transformation and the original weight function* w_k, and $\tilde{v}_k$ has a single child which is v_k *with the original transformation from* u *to* v_k *and an identity weight function*. Under this construction, the `pullback`* operator in the original graph can then be realized in the new graph as

1. a `pullback`* operator from v_k to $\tilde{v}_k$ for each k
2. a `pullback`* operator from $\{\tilde{v}_k\}_{k=1}^K$ to u.

To verify this we rewrite (12.4) as

$$\begin{aligned}
\mathbf{f} &= \sum_{i=1}^{K} w_i \mathbf{J}_i^\top (\mathbf{f}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{x}}) + \mathbf{h}_i =: \sum_{i=1}^{K} w_i \tilde{\mathbf{f}}_i + \tilde{\mathbf{h}}_i \\
\mathbf{M} &= \sum_{i=1}^{K} w_i \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i =: \sum_{i=1}^{K} w_i \tilde{\mathbf{M}}_i \\
\mathbf{G} &= \sum_{i=1}^{K} w_i \mathbf{J}_i^\top \mathbf{G}_i \mathbf{J}_i =: \sum_{i=1}^{K} w_i \tilde{\mathbf{G}}_i \\
L &= \sum_{i=1}^{K} w_i L_i =: \sum_{i=1}^{K} w_i \tilde{L}_i
\end{aligned}$$

where we also has $\mathbf{h}_i = \tilde{\mathbf{h}}_i = \tilde{L}_i \nabla_{\mathbf{x}} w_i - (\dot{\mathbf{x}}^\top \nabla_{\mathbf{x}} w_i) \tilde{\mathbf{G}}_i \dot{\mathbf{x}}$. That is, node $\tilde{v}_k$ has the RMP $(\tilde{\mathbf{f}}_i, \tilde{\mathbf{M}}_i)_C^{\mathcal{M}}$, the metric matrix $\tilde{\mathbf{G}}_i$, and the Lagrangian $\tilde{L}_i$. From the equalities above, we verify the two-step decomposition of `pullback*` is valid.

Next we show that each step in the two-step decomposition yields a structured GDS like Lemma 12.A.1, which is sufficient condition we need to prove Theorem 12.3.1. In the first step from v_i to $\tilde{v}_i$, because the weight is constant identity, `pullback*` is the same as `pullback`. We apply Lemma 12.A.1 and conclude that $\tilde{v}_i$ follows $(\mathcal{M}, \tilde{\mathbf{G}}_i, \tilde{\mathbf{B}}_i, \tilde{\Phi}_i)_{\tilde{\mathcal{S}}_i}$, where $\tilde{\mathcal{S}}_i$ preserves $\mathcal{S}_i$.

Then we show the second step from $\{\tilde{v}_i\}_{i=1}^K$ to u has similar properties. This is summarized as Lemma 12.A.2 below.

Lemma 12.A.2. *If $\tilde{v}_i$ follows $(\mathcal{M}, \tilde{\mathbf{G}}_i, \tilde{\mathbf{B}}_i, \tilde{\Phi}_i)_{\tilde{S}_i}$, then u follows $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_S$, where S preserves $\tilde{S}_i$, $\mathbf{G} = \sum_{i=1}^K w_i \tilde{\mathbf{G}}_i$, $\mathbf{B} = \sum_{i=1}^K w_i \tilde{\mathbf{B}}_i$, and $\Phi = \sum_{i=1}^K w_i \tilde{\Phi}_i$.*

Proof of Lemma 12.A.2. This can be shown by algebraically comparing the dynamics of $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_S$ and the result of (12.4). Let $\mathbf{x}$ be a coordinate of $\mathcal{M}$ and, without loss of generality, let us consider w_k to be a function of only $\mathbf{x}$ (we ignore the dependency on the auxiliary state). By Definition 12.A.1, the dynamics of $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_S$ satisfies

$$\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) \ddot{\mathbf{x}} + \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}}) \dot{\mathbf{x}} \tag{12.8}$$

We first show the recursion of $\mathbf{f}$ of `pullback`* satisfies (12.8). To this end, we rewrite $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}$ by Definition 12.A.1 as

$$\begin{aligned}\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x},\dot{\mathbf{x}}) &= \sum_{i=1}^{K} \boldsymbol{\xi}_{w_i\tilde{\mathbf{G}}_i}(\mathbf{x},\dot{\mathbf{x}}) \\ &= \sum_{i=1}^{K} w_i(\mathbf{x})\boldsymbol{\eta}_{\tilde{\mathbf{G}}_i}(\mathbf{x},\dot{\mathbf{x}}) + (\dot{\mathbf{x}}^\top \nabla_{\mathbf{x}} w_i(\mathbf{x}))\tilde{\mathbf{G}}_i(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}} - \frac{1}{2}\nabla_{\mathbf{x}} w_i(\mathbf{x})\dot{\mathbf{x}}^\top \tilde{\mathbf{G}}_i(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}}\end{aligned}$$

where in the first equality we use the trick we made that the transformation from u to $\tilde{v}_k$ is identity and we use the fact $\tilde{S}_i$ preserves S_i, so the structure S that preserves $\tilde{S}_i$ has a clean structure

$$\mathbf{G} = \begin{bmatrix} I & \dots & I \end{bmatrix} \begin{bmatrix} w_1\tilde{\mathbf{G}}_1 & & \\ & \ddots & \\ & & w_K\tilde{\mathbf{G}}_K \end{bmatrix} \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}$$

Similarly, we rewrite $\nabla_{\mathbf{x}}\Phi(\mathbf{x}) = \sum_{i=1}^{K} w_i(\mathbf{x})\nabla_{\mathbf{x}}\tilde{\Phi}_i(\mathbf{x}) + \tilde{\Phi}_i\nabla_{\mathbf{x}} w_i(\mathbf{x})$. Substituting these two equalities into (12.8), we can write (with input dependency omitted)

$$\begin{aligned}&\mathbf{M}\ddot{\mathbf{x}} = -\nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}} - \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}} \\ &= \sum_{i=1}^{K} -w_i\nabla_{\mathbf{x}}\tilde{\Phi}_i - \tilde{\Phi}_i\nabla_{\mathbf{x}} w_i - w_i\mathbf{B}_i\dot{\mathbf{x}} + \sum_{i=1}^{K} -w_i\boldsymbol{\eta}_{\tilde{\mathbf{G}}_i} - (\dot{\mathbf{x}}^\top\nabla_{\mathbf{x}} w_i)\tilde{\mathbf{G}}_i\dot{\mathbf{x}} + \frac{1}{2}\nabla_{\mathbf{x}} w_i\dot{\mathbf{x}}^\top\tilde{\mathbf{G}}_i\dot{\mathbf{x}} \\ &= \sum_{i=1}^{K} w_i\tilde{\mathbf{f}}_i + \frac{1}{2}\nabla_{\mathbf{x}} w_i\dot{\mathbf{x}}^\top\tilde{\mathbf{G}}_i\dot{\mathbf{x}} - \tilde{\Phi}_i\nabla_{\mathbf{x}} w_i - (\dot{\mathbf{x}}^\top\nabla_{\mathbf{x}} w_i)\tilde{\mathbf{G}}_i\dot{\mathbf{x}} \\ &= \sum_{i=1}^{K} w_i\tilde{\mathbf{f}}_i + \mathbf{h}_i\end{aligned}$$

where we use the fact that $\tilde{\mathbf{f}}_i = -\nabla_{\mathbf{x}}\tilde{\Phi}_i - \tilde{\mathbf{B}}_i\dot{\mathbf{x}} - \boldsymbol{\eta}_{\tilde{\mathbf{G}}_i;\tilde{\mathcal{S}}_i}$ as $\tilde{v}_i$ follows $(\mathcal{M}, \tilde{\mathbf{G}}_i, \tilde{\mathbf{B}}_i, \tilde{\Phi}_i)_{\tilde{\mathcal{S}}_i}$ with $\tilde{\mathcal{S}}_i$ preserving $\mathcal{S}_i$. This is exactly the recursion of $\mathbf{f}$ when `pullback`* is applied between $\tilde{v}_i$ and u, i.e. $\mathbf{f} = \mathbf{M}\ddot{\mathbf{x}} = \sum_{i=1}^{K} w_i\tilde{\mathbf{f}}_i + \mathbf{h}_i$.

To establish the equivalence of the other recursions, we next rewrite $\mathbf{M}$ by definition

in (13.3) as

$$\begin{aligned}\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) &= \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \\ &= \sum_{i=1}^{K} w_i(\mathbf{x}) \left(\tilde{\mathbf{G}}_i(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\tilde{\mathbf{G}}_i}(\mathbf{x}, \dot{\mathbf{x}}) \right) \\ &= \sum_{i=1}^{K} w_i(\mathbf{x}) \tilde{\mathbf{M}}_i(\mathbf{x}, \dot{\mathbf{x}})\end{aligned}$$

where we use the fact that w_i does not on the velocity $\dot{\mathbf{x}}$. The recursion for $\mathbf{G}$ and L can be derived similarly, so we omit them here. ■

So far we have shown that `pullback*` of RMPfusion retains the closure of structured GDSs as `pullback` in RMPflow. In addition, we show that the structured GDS created by `pullback*` has a linearly weighted metric matrix, damping matrix, and potential function (cf. Lemma 12.A.2). By recursively applying the two-step decomposition above, from the leaf nodes to the root node, we conclude that the root node policy will be a structured GDS with a Lyapunov function given by the recursion in (12.5). The rest of the statement of Theorem 12.3.1 follows from the properties of structured GDSs as shown in (Cheng et al., 2018b).

12.B Benefits due to the Extra Flexibility of RMPfusion

We use an example to illustrate the extra flexibility offered by RMPfusion. Consider a simple Y-shape RMP-tree* with a root node and two child nodes with weight functions w_1 and w_2. For the child nodes, suppose they are GDS $(\mathcal{N}_i, \mathbf{G}_i, \mathbf{B}_i, \Phi_i)$ and have coordinate $\mathbf{y}_i$, for $i = 1, 2$. For simplicity, let us assume $\mathbf{G}_i$ only depends on the configuration $\mathbf{y}_i$. From Theorem 12.3.1, we see that the root node has an energy function $V_{\mathrm{r}} = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{G}_r \dot{\mathbf{q}} + \Phi_r$, where $\mathbf{G}_r(\mathbf{q}) = w_1(\mathbf{q})\mathbf{G}_1(\mathbf{y}_1(\mathbf{q})) + w_2(\mathbf{q})\mathbf{G}_2(\mathbf{y}_2(\mathbf{q}))$ and $\Phi_r(\mathbf{q}) = w_1(\mathbf{q})\Phi_1(\mathbf{y}_1(\mathbf{q})) + w_2(\mathbf{q})\Phi_2(\mathbf{y}_2(\mathbf{q}))$ Because w_i is a function of $\mathbf{q}$ not $\mathbf{y}_i$ and the Lyapunov function of RMPflow only allows summing child-node functions, this example root node

policy does not admit a tree structure decomposition in the original RMP-tree and can only be implemented as a single large node. Conversely, because of the weight function on the edges, RMP-tree* can further exploits potential sparsity inside the policy representation so that building complicated global polices with only basic elementary policies becomes possible.

We note that the example above does not imply that RMPfusion can generate more expressive policies than RMPflow. More precisely, RMP-tree* allows representing the same global policy using more basic leaf-node policies. This property has two implications: it suggests (i) RMPfusion can be more efficient to compute and (ii) RMPfusion can offload the difficulties of designing leaf-nodes policies into the weight functions, which are learnable.

12.C Learning RMPfusion

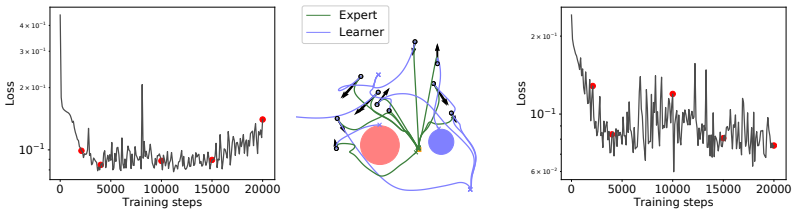
To show the weights are learnable, it is sufficient to check if we can differentiate through the output of the final policy $\pi = \mathbf{a}_{\mathrm{r}}$ with respect to the parameters that specify the weight functions. As the computation of $\mathbf{a}_{\mathrm{r}}$ is accomplished recursively in the backward pass using `pullback`*, we will only illustrate that `pullback`* is differentiable. This can be seen by treating `pullback`* as a computation graph, as illustrated in Figure 12.2. Take the nodes in (12.4) as an example. `pullback`* receives $\mathbf{f}_i$, $\mathbf{M}_i$, $\mathbf{G}_i$, $\mathbf{B}_i$, $\mathbf{J}_i$, $\dot{\mathbf{J}}_i$, L_i from the edges to the child nodes, the current state $(\mathbf{x}, \dot{\mathbf{x}})$ and the auxiliary state to define the weight function w_i and the correction term $\mathbf{h}_i$. As these inputs values do not depend on the weight functions $\{w_i\}$ at the current node (i.e. they do not form a loop), the derivative of $\mathbf{a}_r$ with respect to the weight functions in the RMP-tree* can be computed recursively by back-propagating the derivatives through each `pullback`* operator.

12.D Experimental Details

12.D.1 2D Robot

`2d1level` consists of a 2D particle that aims to reach a goal while avoiding an obstacle. The RMP-tree* for `2d1level` is of depth one (see Figure 12.2b), where the root node q (configuration space of the robot) has one child obstacle RMP node (o_{rmp}) and one child attractor RMP node (a_{rmp}). `2d2level` consists of a 2D particle that aims to reach a goal while avoiding two obstacles. The RMP-tree* for `2d2level` is of depth two (see Fig 12.2c), where the root node (q) has one child attractor RMP node (a_{rmp}) and one all-obstacle RMP (o) that is meant to combine two child obstacle RMPs (o_{rmp}, one for each obstacle). The respective weight functions are shown on the edges of both these trees. The tree structures here are heuristically chosen based on the problem domain, as in RMPflow and typically follow the robots kinematic chain and then extend into the workspace and abstract task spaces.

Figure 12.7 shows the progression of `learner-un` during training, in which each snapshot corresponds to an associated point on the training curve in Figure 12.6a. We see that `learner-un` is never able to reach the goal and often ends up in collision during and after training. We also compared with a unstructured network, `learner-un-large`, that has 5.8 times more learnable parameters compared to `learner-un`. We see improvement over loss values where the batch-loss is 0.065 and the online-loss is 0.393, and the collision rate decreases to 16%. However, it is still never able to complete the task (e.g. see Figure 12.6b). Figure 12.8 shows the progression of `learner-un-large` during training, in which each snapshot corresponds to an associated point on the training curve in Figure 12.6c.



(a) (b) (c)

Figure 12.6: (b) Trajectories generated in `2d2level` by `learner-rmp-large` compared to the expert is shown. Initial state is a black circle for position and black arrow for velocity. The environment has obstacles (red and blue) and goal (orange square). Learning curves for (a) `learner-rmp` and (c) `learner-rmp-large` on `2d2level` is also shown.

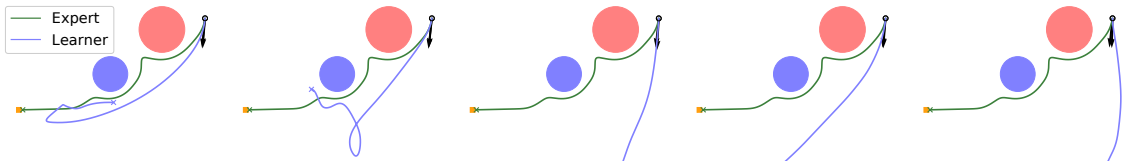


Figure 12.7: Trajectories produced by `learner-un` at various stages during training for `2d2level`. From left to right these plots correspond to the red dots from left to right on the training curve in Figure 12.6a.

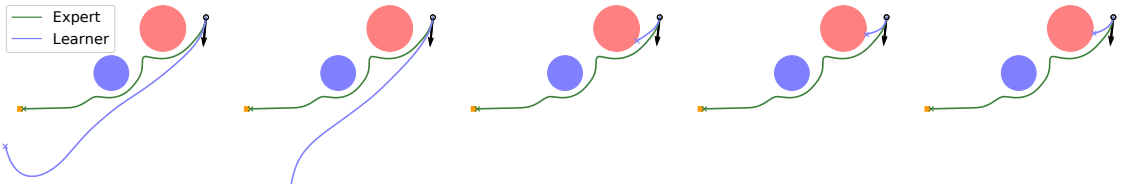


Figure 12.8: Trajectories produced by `learner-un-large` at various stages during training for `2d2level`. From left to right these plots correspond to the red dots from left to right on the training curve in Figure 12.6c.

12.D.2 Franka Robot

From the root node we have various task spaces, like the end-effector position (ee) on which the attractor space (a) is defined by a change of coordinates such that the goal position is at the origin. The attractor RMP (a_{rmp}) is then defined on the attractor space for a goal reaching subtask. Each joint of the robot is mapped to a one dimensional upper (ujl_i) and lower (ljl_i) joint limit space where a joint limit RMP (jl_{rmp}) is defined for joint limit avoidance subtasks. The root node is also mapped to a pre-specified number of control points on the robot (cp_i) such that they collectively approximate the robot's body and can

be used for collision avoidance. On any control point space we add a distance space to the obstacle (d_i) where the obstacle RMP (o_{rmp}) is defined. Note that when multiple obstacles are present we can add distance spaces and the obstacle RMPs for each obstacle on every control point. Now, since the tree structure can change with the number of obstacles, in practice, shared weights can be specified across all obstacles on a given control point, such that training can be performed with only one obstacle to learn the weight function and then can be applied to arbitrary number of obstacles during execution. Finally, there are also native RMPs defined on the root node like a constant damper RMP (q_d) and an RMP which is just an identity metric (q_{mi}) with no learnable weight function to ensure the `resolve` operator is numerically stable.

Figure 12.9 shows a qualitative comparison on an example execution with the expert and the learners. We verify the stability properties of RMPfusion (even during learning) with the monotonically decreasing Lyapunov function plots on these executions. Note that the scale on the plot for `learner-0` is very small and the tiny kink on the plot is due to numerical issues with Euler integration.

12.D.3 Discussion

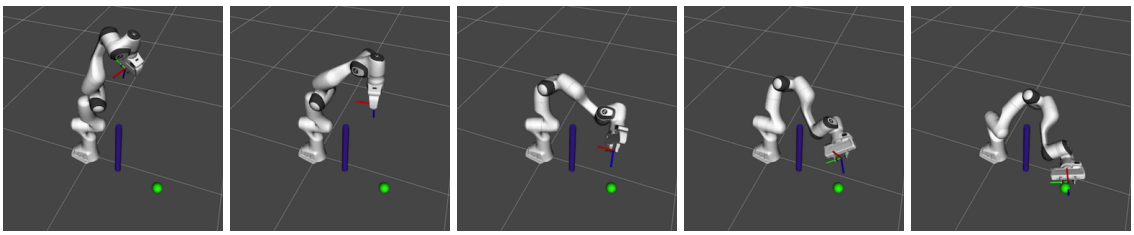
The experiments shown here were designed to study if RMPfusion can combine imperfect subtask RMPs, whose inertia weight functions are incorrectly specified while motion policies are sensibly designed with domain knowledge. While this setup does not emulate the full generality where everything is unknown, we think that it captures a representative and important scenario that often happens in practice. Weve had extensive literature in designing motion policies, whereas designing the associated metrics/inertias for these policies is a fairly new and nontrivial concept, which is a major user burden imposed by RMPflow.

We address this issue by learning the weight functions, and show in the experiments that imperfect subtask RMPs with poorly designed metrics can still be compensated by our framework. Importantly, we emphasize that RMPfusion is designed for generality and does

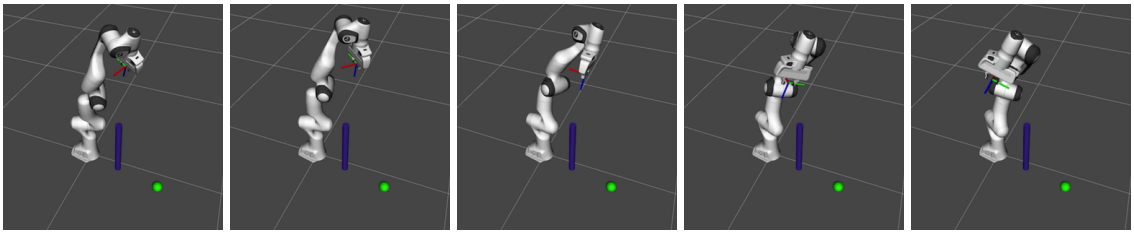
not assume the knowledge that only the inertias are wrongly specified. Therefore, though not tested in the current experiments, we do believe RMPfusion can be used in more general setups, so long as the user provides sufficiently rich subtask RMPs such that there exists a fusion that can generate the desired behavior. However, how to choose the subtask RMPs to start with is a domain specific problem, similar to specifying the size and structure of a neural network in general. Therefore, we consider it beyond the scope of the current paper, because our main focus here is to study and validate the theoretical benefits of RMPfusion (like stability during immature learning).

Generally, an RMPfusion policy with constant weights (not a function of the parent state, etc.) can be reduced into an RMPflow policy with the same tree structure. This can be seen from (12.4); when the weights are constant, we can effectively push all the weights of an RMP-tree* to the leaf-nodes to define modified inertia matrices on an RMP-tree (the motion policy doesn't change). In other words, in the experiments, the expert can be viewed as an RMPflow policy with some unknown inertia matrices and therefore RMPflow wasn't directly compared.

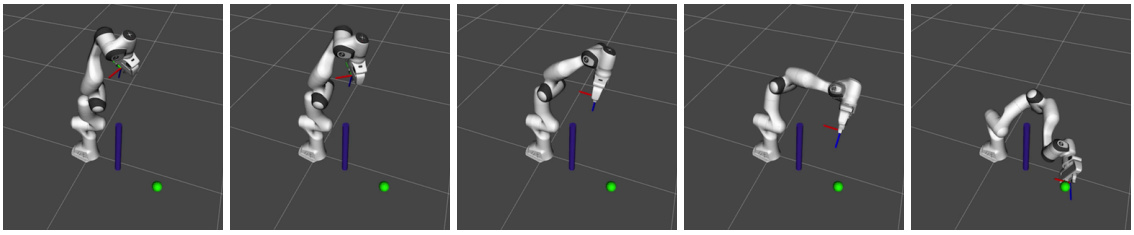
Using neural networks to parameterize the weight functions maybe is an overkill in our experiments. The reason for using general function approximators here is to show that our framework is practically feasible and can support situations where this will become necessary. For example, this allows for learning general differentiable representation for the weight functions, e.g., using images for auxiliary states. However, one should note also that while using expressive function approximators would add representation to the whole policy it could also potentially make learning more difficult.



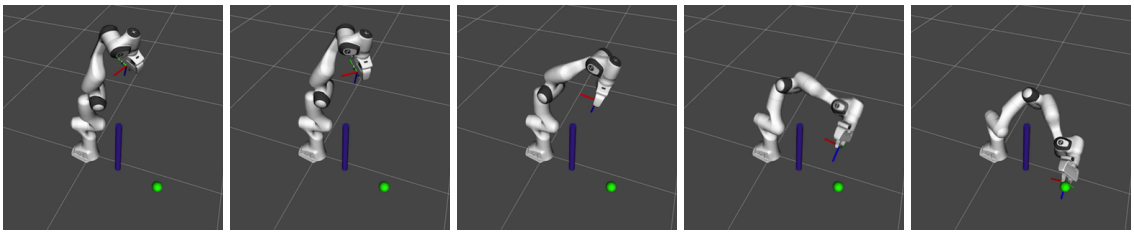
(a) expert



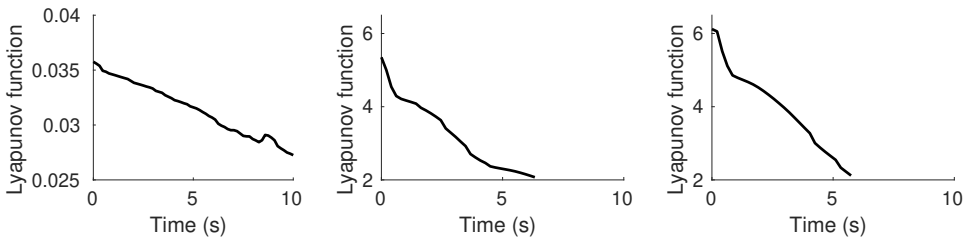
(b) `learner-0`



(c) `learner-300`



(d) `learner-1200`



(e)

Figure 12.9: (a)-(d) An example execution (left to right) from the test dataset, comparing (a) the expert with (b) `learner-0`, (c) `learner-300`, and (d) `learner-1200`. (e) The respective Lyapunov function of the learners' trajectories (`learner-0` (left), `learner-300` (middle), `learner-1200` (right)).

CHAPTER 13
RMPFLOW WITH CONTROL LYAPUNOV FUNCTION

13.1 Introduction

Multi-objective tasks are often involved in the control of robotic systems (Morris, Powell, and Ames, 2013; Peters et al., 2008; Ratliff, Issac, and Kappler, 2018; Wang, Ames, and Egerstedt, 2016). For example, a group of robots may be tasked with achieving a certain formation, moving toward a goal region, while avoiding collisions with each other and obstacles (Wang, Ames, and Egerstedt, 2016). These types of problems call for algorithms that can systematically generate a stable controller capable of fulfilling multiple control specifications simultaneously.

A classic strategy is to first design a controller for each individual control specification, and then provide a high-level rule to *switch* among them. This idea has been frequently exploited in robotics (Arkin, 1998). For example, it is common practice to switch to a collision avoidance controller when the robot risks colliding with obstacles (Arkin, 1998). The stability of switching systems has been thoroughly investigated, e.g. by finding a common or switched Lyapunov function for the systems among all designed controllers (Daafouz, Riedinger, and Iung, 2002; Liberzon, Hespanha, and Morse, 1999; Narendra and Balakrishnan, 1994; Vu and Liberzon, 2005). However, a fundamental limitation shared by these switching approaches is that only a single controller is active at a time and hence only a subset of the control specifications is considered. If not designed properly, some controllers for secondary tasks might take over the operation for most of the time. For example, when a robot navigates in a cluttered environment, the collision avoidance controller can dominate other controllers and the primary tasks may never be considered (Wang, Ames, and Egerstedt, 2016). Therefore, it may be more desirable to *blend* controllers rather than impose a

hard *switch* between them, so that all tasks can be considered simultaneously.

In robotics, the strategy of weighting controllers for different tasks has been explored in potential field methods (Arkin, 1998; Khatib, 1985). While easy to implement such schemes, it can be difficult to provide formal stability guarantees for the overall "blended" system, especially when the weights are state-dependent. In some cases, the stability of the overall system has been shown through a common Lyapunov function (Narendra and Balakrishnan, 1994; Vu and Liberzon, 2005), but the existence of a common Lyapunov function is not guaranteed. Finding a common Lyapunov function can be particularly challenging for robotics applications because the tasks can potentially conflict, e.g. the robot may need to move through a cluttered environment to go to the goal.

The framework of null-space or hierarchical control handles this problem by assigning priorities to the tasks, and hence to the controllers (Escande, Mansard, and Wieber, 2014; Peters et al., 2008). The performance of the high-priority tasks can be guaranteed by forcing the lower-priority controllers to act on the null space of high-priority tasks. However, several problems surface as the number of tasks increases. One problem is the algorithmic singularities introduced by the usage of multiple levels of projections (Escande, Mansard, and Wieber, 2014; Peters et al., 2008). Most algorithms are designed under the assumption of singular-free conditions. But this assumption is unlikely to hold in practice, especially when there are a large number of tasks, and the system can easily become unstable if the algorithmic singularities occur. In addition, similar to the switching scheme, it is possible that secondary controllers, e.g. collision avoidance controllers, become the ones with high-priorities and the primary task can not be achieved. While several heuristics (Dietrich, Albu-Schäffer, and Hirzinger, 2012; Lee, Mansard, and Park, 2012) have been proposed to shift the control priorities dynamically, whether such systems can be globally stabilized in presence of the algorithmic singularities is still an open question (Dietrich, Ott, and Park, 2018).

Control Lyapunov functions (CLFs) and control barrier functions (CBFs) constitute

another class of methods to encode multiple control specifications (Ames, Grizzle, and Tabuada, 2014; Morris, Powell, and Ames, 2013; Wang, Ames, and Egerstedt, 2016). In the CLF and CBF frameworks, the control specifications are encoded as constraints on the time derivatives of Lyapunov or barrier function candidates, and a control input that satisfies all the constraints is solved through a constrained optimization problem. However, in the case of conflicting specifications, the CLF and CBF frameworks suffer from feasibility problems (Squires, Pierpaoli, and Egerstedt, 2018), i.e. there does not exist any controller that satisfies all the control specifications. Although the CLF constraints can be relaxed through slack variables (Ames, Grizzle, and Tabuada, 2014), they also add a new set of hyperparameters to trade off the importance of different specifications; care must be taken in tuning these hyperparameters in order to achieve desired performance properties and maintain stability. Finally, it can be hard to encode certain high-dimensional control specifications, such as damping behaviors, as CLF or CBF constraints.

In this chapter, we focus on *weighting* individual controllers. We aim to address two interrelated questions:

- How can controllers be composed while guaranteeing system stability?
- How should individual controllers be designed so that they can be easily combined?

Although ensuring stability is challenging for arbitrary blending schemes, we design a systematic process to combine controllers so that the stability of the overall system is guaranteed. Our framework considers all control specifications simultaneously, while providing the flexibility to vary the importance of different controllers based on the robot state. Moreover, instead of considering specifications in the configuration space, we allow for controllers defined directly on different spaces or manifolds[1] for different specifications.

This separation can largely simplify the design and computation of each individual controller, because it only concerns a possibly lower-dimensional manifold that is directly

[1]Specifications defined on non-Euclidean manifolds are common in robotics; for example, in obstacle avoidance, obstacles become holes in the space and the geodesics flow around them (Ratliff, Issac, and Kappler, 2018).

relevant to a particular control specification. For example, controllers for different links of a robot manipulator can be designed in their corresponding (possibly non-Euclidan) workspaces. We leverage a recent approach to controller synthesis in robotics, the Riemannian Motion Policies (RMPs) (Ratliff, Issac, and Kappler, 2018) and RMPflow (Cheng et al., 2018b) introduced in Chapter 11, which have been successfully deployed on robot manipulators (Cheng et al., 2018b; Ratliff, Issac, and Kappler, 2018) and multi-robot systems (Li et al., 2019a). An RMP is a mathematical object that is designed to describe a controller on a manifold, and RMPflow is a computational framework for combining RMPs designed on different task manifolds into a controller for the entire system. A particular feature of RMPflow is the use of state-dependent importance weightings of controllers based on the properties of the corresponding manifolds. It is show in (Cheng et al., 2018b) that when RMPs are generated from Geometric Dynamical Systems (GDSs), the combined controller is Lyapunov-stable.

In Chapter 11, we studied RMPs and RMPflow in terms of the geometric structure of second-order differential equations (Cheng et al., 2018b), where Riemannian metrics on manifolds (of GDSs) naturally provide a geometrically-consistent notion of task importance and hence a mechanism to combine controllers (i.e. RMPflow). While differential geometry provides a mathematical interpretation of RMPflow, in practice, the restriction to GDSs for control specifications could limit performance and make controller design difficult.

To overcome this limitation, here we revisit RMPflow with a rigorous CLF treatment and show that the existing computational framework of RMPflow actually ensures stability for a *larger* class of systems than GDSs. This discovery is made possible by an alternative stability analysis of RMPflow and an induction lemma that characterizes how the stability of individual controllers is propagated to the combined controller in terms of CLF constraints. Hence, we can reuse RMPflow to stably combine a range of controllers, not limited to the ones consistent with GDSs. To demonstrate, we introduce a computational

framework called *RMPflow–CLF*, where we augment RMPflow with CLF constraints to generate a stable controller given user-specified nominal controllers for each of the control specifications. This allows users to incorporate additional design knowledge given by, e.g. heuristics, motion planners, and human demonstrations, without worrying about the geometric properties of the associated manifolds. RMPflow–CLF can be viewed as a soft version of the QP–CLF framework (Morris, Powell, and Ames, 2013) that guarantees the stability of the overall system, while ensuring feasibility even when control specifications are conflicting. This chapter is partly based on our paper published as (Li et al., 2019b).

13.2 Background

For convenience of reading, we first shortly review RMPflow (Cheng et al., 2018b; Ratliff, Issac, and Kappler, 2018) and CLFs (Ames, Grizzle, and Tabuada, 2014; Morris, Powell, and Ames, 2013), which are different ways to combine control specifications.

13.2.1 Riemannian Motion Policies (RMPs) and RMPflow

Consider a robot with configuration space $\mathcal{C}$ which is a smooth d-dimensional manifold. We assume that $\mathcal{C}$ admits a global *generalized coordinate* $\mathbf{q} : \mathcal{C} \to \mathbb{R}^d$ and follow the assumption in (Cheng et al., 2018b) that the system can be feedback linearized in such a way that it can be controlled directly through the generalized acceleration[2], i.e. $\ddot{\mathbf{q}} = \mathbf{u}(\mathbf{q}, \dot{\mathbf{q}})$. We call $\mathbf{u}$ a *control policy* or a *controller*, and $(\mathbf{q}, \dot{\mathbf{q}})$ the *state*.

The task is often defined on a different manifold from $\mathcal{C}$ called the *task space*, denoted $\mathcal{T}$. A task may admit further structure as a composition of *subtasks* (e.g. reaching a goal, avoiding collision with obstacles, etc.). In this case, we can treat the task space as a collection of multiple lower-dimensional *subtask spaces*, each of which is a manifold. In other words, each subtask space is associated with a control specification and together the task space $\mathcal{T}$ describes the overall multi-objective control problem.

[2]This setup can be extended to torque controls as in (Peters et al., 2008).

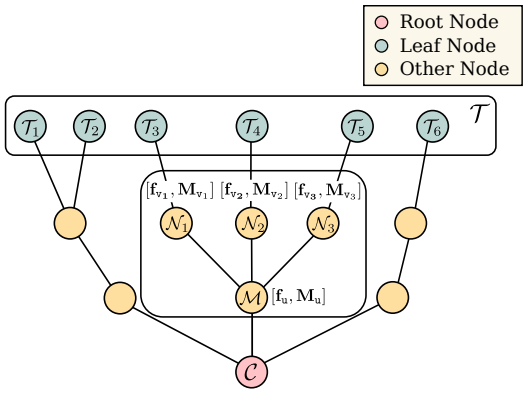


Figure 13.1: An example of an RMP-tree. See text for details.

Ratliff et al. (Ratliff, Issac, and Kappler, 2018) propose *Riemannian Motion Policies* (RMPs) to represent control policies on manifolds. Consider an m-dimensional manifold $\mathcal{M}$ with a global coordinate $\mathbf{x} \in \mathbb{R}^m$. An RMP on $\mathcal{M}$ can be represented by two forms, its *canonical form* $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ and its *natural form* $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$, where $\mathbf{a} : (\mathbf{x}, \dot{\mathbf{x}}) \mapsto \mathbf{a}(\mathbf{x}, \dot{\mathbf{x}})$ is the desired acceleration, $\mathbf{M} : (\mathbf{x}, \dot{\mathbf{x}}) \mapsto \mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}^{m \times m}_{+}$ is the inertial matrix, and $\mathbf{f} = \mathbf{M}\mathbf{a}$ is the desired force. It is important to note that $\mathbf{M}$ and $\mathbf{f}$ do not necessarily correspond to physical quantities; $\mathbf{M}$ defines the importance of an RMP when combined with other RMPs, and $\mathbf{f}$ is proposed for computational efficiency.

RMPflow (Cheng et al., 2018b) is a recursive algorithm to generate control policies on the configuration space given the RMPs of subtasks. It introduces: 1) a data structure, the *RMP-tree*, for computational efficiency; and 2) a set of operators, the *RMP-algebra*, to propagate information across the RMP-tree.

An RMP-tree is a directed tree, which encodes the computational structure of the task map from $\mathcal{C}$ to $\mathcal{T}$ (see Fig. 13.1). In the RMP-tree, a node is associated with the state and the RMP on a manifold, and an edge is augmented with a smooth map from a parent-node manifold to a child-node manifold. In particular, the root node $\mathtt{r}$ is associated with the state of the robot $(\mathbf{q}, \dot{\mathbf{q}})$ and its control policy on the configuration space $(\mathbf{a}_{\mathtt{r}}, \mathbf{M}_{\mathtt{r}})^{\mathcal{C}}$, and each leaf node $\mathtt{l}_k$ is associated with the RMP $(\mathbf{a}_{\mathtt{l}_k}, \mathbf{M}_{\mathtt{l}_k})^{\mathcal{T}_k}$ for a subtask, where $\mathcal{T}_k$ is a subtask manifold. Recall the collection $\{\mathcal{T}_k\}_{k=1}^{K}$ is the task space $\mathcal{T}$, where K is the number of

tasks.

To illustrate how the RMP-algebra operates, consider a node $\mathtt{u}$ with N child nodes $\{\mathtt{v}_j\}_{j=1}^N$. Let $\mathtt{e}_j$ denote the edge from $\mathtt{u}$ to $\mathtt{v}_j$ and let $\psi_{\mathtt{e}_j}$ be the associated smooth map. Suppose that $\mathtt{u}$ is associated with an RMP $[\mathbf{f}_\mathtt{u}, \mathbf{M}_\mathtt{u}]^\mathcal{M}$ on a manifold $\mathcal{M}$ with coordinate $\mathbf{x}$, and $\mathtt{v}_j$ is associated with an RMP $[\mathbf{f}_{\mathtt{v}_j}, \mathbf{M}_{\mathtt{v}_j}]^{\mathcal{N}_j}$ on a manifold $\mathcal{N}_j$ with coordinate $\mathbf{y}_j$. (Note that here we represent the RMPs in their natural form.) The RMP-algebra consists of the following three operators:

1. `pushforward` is the operator to forward propagate the *state* from the parent node $\mathtt{u}$ to its child nodes $\{\mathtt{v}_j\}_{j=1}^N$. Given the state $(\mathbf{x}, \dot{\mathbf{x}})$ from $\mathtt{u}$, it computes $(\mathbf{y}_j, \dot{\mathbf{y}}_j) = (\psi_{\mathtt{e}_j}(\mathbf{x}), \mathbf{J}_{\mathtt{e}_j}(\mathbf{x})\,\dot{\mathbf{x}})$ for each child node v_j, where $\mathbf{J}_{\mathtt{e}_j} = \partial_\mathbf{x}\psi_{\mathtt{e}_j}$ is the Jacobian matrix of $\psi_{\mathtt{e}_j}$.

2. `pullback` is the operator to backward propagate the RMPs from the child nodes to the parent node. Given $\{[\mathbf{f}_{\mathtt{v}_j}, \mathbf{M}_{\mathtt{v}_j}]^{\mathcal{N}_j}\}_{j=1}^N$ from the child nodes, the RMP $[\mathbf{f}_\mathtt{u}, \mathbf{M}_\mathtt{u}]^\mathcal{M}$ for the parent node $\mathtt{u}$ is computed as,
$$\mathbf{f}_\mathtt{u} = \sum_{j=1}^N \mathbf{J}_{\mathtt{e}_j}^\top(\mathbf{f}_{\mathtt{v}_j} - \mathbf{M}_{\mathtt{v}_j}\dot{\mathbf{J}}_{\mathtt{e}_j}\dot{\mathbf{x}}),\ \mathbf{M}_\mathtt{u} = \sum_{j=1}^N \mathbf{J}_{\mathtt{e}_j}^\top \mathbf{M}_{\mathtt{v}_j}\mathbf{J}_{\mathtt{e}_j}.$$

3. `resolve` maps an RMP from its natural form to its canonical form. Given $[\mathbf{f}_\mathtt{u}, \mathbf{M}_\mathtt{u}]^\mathcal{M}$, it outputs $(\mathbf{a}_\mathtt{u}, \mathbf{M}_\mathtt{u})^\mathcal{M}$ with $\mathbf{a}_\mathtt{u} = \mathbf{M}^\dagger\,\mathbf{f}_\mathtt{u}$, where $\dagger$ denotes Moore-Penrose inverse.

RMPflow performs control policy generation through running the RMP-algebra on the RMP-tree. It first performs a forward pass, by recursively calling `pushforward` from the root node to the leaf nodes to update the state associated with each node on the RMP-tree. Second, every leaf node $\mathtt{l}_k$ *evaluates* its natural form RMP $\{(\mathbf{f}_{\mathtt{l}_k}, \mathbf{M}_{\mathtt{l}_k})^{\mathcal{T}_{\mathtt{l}_k}}\}_{k=1}^K$ given its associated state. Then, RMPflow performs a backward pass, by recursively calling `pullback` from the leaf nodes to the root node to back propagate the RMPs in the natural form. Finally, `resolve` is applied to the root node to transform the RMP $[\mathbf{f}_\mathtt{r}, \mathbf{M}_\mathtt{r}]^\mathcal{C}$ into its canonical form $(\mathbf{a}_\mathtt{r}, \mathbf{M}_\mathtt{r})^\mathcal{C}$ and set the control policy as $\mathbf{u} = \mathbf{a}_\mathtt{r}$.

RMPflow was originally analyzed based on a differential geometric interpretation. Cheng et al. (Cheng et al., 2018b) consider the inertial matrix $\mathbf{M}$ generated by a Riemannian metric on the *tangent bundle* of the manifold $\mathcal{M}$ (denoted as $T\mathcal{M}$). Let $\mathbf{G} : (\mathbf{x}, \dot{\mathbf{x}}) \mapsto \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}_+^{m\times m}$ be a (projected) Riemannian metric and define the *curvature terms*

$$\begin{aligned} \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) &\coloneqq \frac{1}{2}\sum_{i=1}^{m} \dot{x}_i \, \partial_{\dot{\mathbf{x}}} \, \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}}), \\ \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) &\coloneqq \overset{\mathbf{x}}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\, \dot{\mathbf{x}} - \frac{1}{2}\nabla_{\mathbf{x}} \left(\dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\, \dot{\mathbf{x}}\right), \end{aligned} \tag{13.1}$$

where $\overset{\mathbf{x}}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq [\partial_{\mathbf{x}} \, \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})\, \dot{\mathbf{x}}]_{i=1}^{m}$, $\mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})$ is the ith column of $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$, and x_i is the ith component of $\mathbf{x}$. The inertial matrix $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ is then related to $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ through,

$$\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}). \tag{13.2}$$

Under this geometric interpretation, RMPs on a manifold $\mathcal{M}$ can be (but not necessarily) generated from a class of systems called *Geometric Dynamical Systems* (GDSs) (Cheng et al., 2018b), whose dynamics are on the form of

$$\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})\, \ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\, \dot{\mathbf{x}}, \tag{13.3}$$

where $\mathbf{B} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m\times m}$ is the *damping matrix*, and $\Phi : \mathbb{R}^m \to \mathbb{R}$ is the *potential function*. When $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$, the GDSs reduce to the widely studied *Simple Mechanical Systems* (Bullo and Lewis, 2004).

The stability properties of RMPflow is analyzed in (Cheng et al., 2018b) under the assumption that *every* leaf-node RMP is specified as a GDS (13.3). Before stating the stability theorem, let us define the metric, damping matrix, and potential function for every node in the RMP-tree: For a leaf node, its metric, damping matrix, and potential are defined naturally by its underlying GDS. For a non-leaf node $\mathtt{u}$ with N children $\{\mathtt{v}_j\}_{j=1}^{N}$, these

terms are defined recursively by the relationship,

$$\mathbf{G}_{\mathtt{u}} = \sum_{j=1}^{N} \mathbf{J}_{\mathtt{e}_j}^{\top} \mathbf{G}_{\mathtt{v}_j} \mathbf{J}_{\mathtt{e}_j}, \; \mathbf{B}_{\mathtt{u}} = \sum_{j=1}^{N} \mathbf{J}_{\mathtt{e}_j}^{\top} \mathbf{B}_{\mathtt{v}_j} \mathbf{J}_{\mathtt{e}_j}, \; \Phi_{\mathtt{u}} = \sum_{j=1}^{N} \Phi_{\mathtt{v}_j} \circ \psi_{\mathtt{e}_j}, \tag{13.4}$$

where $\mathbf{G}_{\mathtt{v}_j}$, $\mathbf{B}_{\mathtt{v}_j}$ and $\Phi_{\mathtt{v}_j}$ are the metric, damping matrix, and potential function for the jth child. The stability results for RMPflow are stated below.

Theorem 13.2.1. (Cheng et al., 2018b) *Let $\mathbf{G}_{\mathtt{r}}$, $\mathbf{B}_{\mathtt{r}}$, and $\Phi_{\mathtt{r}}$ be the metric, damping matrix, and potential function of the root node defined in (13.4). If each leaf node is given by a GDS, $\mathbf{G}_{\mathtt{r}}, \mathbf{B}_{\mathtt{r}} \succ 0$, and $\mathbf{M}_{\mathtt{r}}$ is non-singular, then the system converges to a forward invariant set $\mathcal{C}_\infty \coloneqq \{(\mathbf{q}, \dot{\mathbf{q}}) : \nabla_{\mathbf{q}} \Phi_{\mathtt{r}} = 0, \dot{\mathbf{q}} = 0\}$.*

13.2.2 Control Lyapunov Functions (CLFs)

Control Lyapunov Function (CLF) methods (Ames, Grizzle, and Tabuada, 2014; Morris, Powell, and Ames, 2013; Sontag, 1983) encode control specifications as Lyapunov function candidates. In these methods, controllers are designed to satisfy the inequality constraints on the time derivative of the Lyapunov function candidates.

Consider a dynamical system in control-affine form,

$$\dot{\boldsymbol{\eta}} \; = \; f(\boldsymbol{\eta}) + g(\boldsymbol{\eta})\, \mathbf{u}, \tag{13.5}$$

where $\boldsymbol{\eta} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ are the state and control input for the system. We assume that f and g are locally Lipschitz continuous, and the system (13.5) is forward complete, i.e. $\boldsymbol{\eta}(t)$ is defined for all $t \geq 0$. For second-order systems considered by RMPflow, we have $\boldsymbol{\eta} = [\mathbf{x}^\top \; \dot{\mathbf{x}}^\top]^\top$,

$$f(\boldsymbol{\eta}) \equiv \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad g(\boldsymbol{\eta}) \equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{13.6}$$

Suppose that a Lypapunov function candidate $V(\boldsymbol{\eta})$ is designed for a control specification. The control input is then required to satisfy a CLF constraint, e.g. $\dot{V} \leq -\alpha(V)$, where $\alpha : \mathbb{R}_+ \to \mathbb{R}_+$ is a locally Lipschitz class $\mathcal{K}$ function (Khalil, 1996) (i.e. α is strictly

increasing and $\alpha(0) = 0$). In the case of control-affine system, the CLF constraint becomes a linear inequality constraint on control input $\mathbf{u}$ given state $\boldsymbol{\eta}$,

$$L_g V(\boldsymbol{\eta})\,\mathbf{u} \leq -L_f V(\boldsymbol{\eta}) - \alpha(V(\boldsymbol{\eta})), \tag{13.7}$$

where $L_f V$ and $L_g V$ are the *Lie derivatives* of V along f and g, respectively.

When there are multiple control specifications, one can design Lyapunov function candidates $\{V_k\}_{k=1}^K$ separately. Then the controller synthesis problem becomes finding a controller that satisfies all the linear inequalities given by the Lyapunov function candidates. Morris et al. (Morris, Powell, and Ames, 2013) propose a computational framework, QP–CLF, that solves for the controller through a Quadratic programming (QP) problem that augments the constraints with a quadratic objective:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \frac{1}{2}\,\mathbf{u}^\top H(\boldsymbol{\eta})\,\mathbf{u} + F(\boldsymbol{\eta})^\top \mathbf{u} \\ \text{s.t.} \quad & L_g V_k(\boldsymbol{\eta})\,\mathbf{u} \leq -L_f V_k(\boldsymbol{\eta}) - \alpha_k(V_k(\boldsymbol{\eta})), \\ & \qquad \forall\, k \in \{1, \ldots, K\}. \end{aligned} \tag{13.8}$$

However, when the specifications are conflicting, it may not be possible to enforce the CLF constraints for all $\{V_k\}_{k=1}^K$ since the optimization problem (13.8) can become infeasible (Morris, Powell, and Ames, 2013). In (Ames, Grizzle, and Tabuada, 2014), Ames et al. introduce *slack variables* $\{\delta_k\}_{k=1}^K$ so that the optimization problem is always feasible. Let $\bar{\mathbf{u}} = [\mathbf{u}^\top \ \delta_1 \ \ldots \ \delta_K]^\top$ denote all decision variables. Then the relaxed optimization problem becomes,

$$\begin{aligned} \min_{\bar{\mathbf{u}}} \quad & \frac{1}{2}\,\bar{\mathbf{u}}^\top \bar{H}(\boldsymbol{\eta})\,\bar{\mathbf{u}} + \bar{F}(\boldsymbol{\eta})^\top \bar{\mathbf{u}} \\ \text{s.t.} \quad & L_g V_k(\boldsymbol{\eta})\,\mathbf{u} \leq -L_f V_k(\boldsymbol{\eta}) - \alpha(V_k(\boldsymbol{\eta})) + \delta_k, \\ & \qquad \forall\, k \in \{1, \ldots, K\}, \end{aligned} \tag{13.9}$$

where $\bar{H}(\boldsymbol{\eta})$ and $\bar{F}(\boldsymbol{\eta})$ encode how the original objective function and the CLF constraints are balanced. However, care must be taken in tuning $\bar{H}(\boldsymbol{\eta})$ and $\bar{F}(\boldsymbol{\eta})$ to achieve desired

performance properties and maintain stability.

13.3 The CLF Interpretation of RMPflow

The goal of this chapter is to combine control policies specified for subtask manifolds into a control policy for the robot with stability guarantees. RMPflow provides a favorable computational framework but its original analysis is limited to subtask control policies generated by GDSs. This assumption is rather unsatisfying, as the users need to encode the control specifications as GDS behaviors. Further, this restriction can potentially limit the performance of the subtasks and result in unnecessary energy consumption by the system. Although empirically RMPflow has been shown to work with non-GDS leaf policies (Cheng et al., 2018b), it is unclear if the overall system is still stable.

In this section, we show that the RMP-algebra actually preserves the stability of a wider range of leaf-node control policies than GDSs. We relax the original GDS assumption in Chapter 11 to a more general CLF constraint on each leaf node, and provide a novel stability analysis of RMPflow. These results allow us to reuse RMPflow for combining a more general class of control policies, which we will demonstrate by combining controllers based on CLF constraints.

13.3.1 An Induction Lemma

In order to establish CLF constraints on leaf nodes that guarantee stability, we first need to understand how the RMP-algebra, especially the `pullback` operator, connects the stability results of the child nodes to the parent node.

Again, let us consider a node $\mathtt{u}$ with N child nodes $\{\mathtt{v}_j\}_{j=1}^N$, in which $\mathtt{u}$ is associated with a manifold $\mathcal{M}$ with coordinate $\mathbf{x}$, and $\mathtt{v}$ is associated with a manifold $\mathcal{N}_j$ with coordinate $\mathbf{y}_j$. In addition, let $\psi_{\mathtt{e}_j} : \mathbf{x} \mapsto \mathbf{y}_j$ be the smooth map between manifolds $\mathcal{M}$ and $\mathcal{N}_j$. We furthur assume that $\psi_{\mathtt{e}_j}$ is surjective, i.e., $\mathcal{N}_j = \psi_{\mathtt{e}_j}(\mathcal{M})$.

Let us associate each child node $\mathtt{v}_j$ with a proper, continuously differentiable and lower-

bounded potential $\Phi_{\mathtt{v}_j}$ on its manifold $\mathcal{N}_j$ along with a continuously differentiable Riemannian metric $\mathbf{G}_{\mathtt{v}_j}$ on its tangent bundle $\mathcal{TN}_j$. Then, for node $\mathtt{v}_j$, there is a natural Lyapunov function candidate,

$$V_{\mathtt{v}_j}(\mathbf{y}_j, \dot{\mathbf{y}}_j) = \frac{1}{2}\, \dot{\mathbf{y}}_j^\top\, \mathbf{G}_{\mathtt{v}_j}(\mathbf{y}_j, \dot{\mathbf{y}}_j)\, \dot{\mathbf{y}}_j + \Phi_{\mathtt{v}_j}(\mathbf{y}_j), \tag{13.10}$$

and an associated natural-formed RMP $[\mathbf{f}_{\mathtt{v}_j}, \mathbf{M}_{\mathtt{v}_j}]^{\mathcal{N}_j}$, where $\mathbf{f}_{\mathtt{v}_j}$ is the force policy, and $\mathbf{M}_{\mathtt{v}_j}$ is defined by $\mathbf{G}_{\mathtt{v}_j}$ as in (13.2). We shall further assume that $\mathbf{M}_{\mathtt{v}_j}$ is locally Lipschitz continuous for the ease of later analysis. By construction of the RMP-algebra, these Lyapunov function candidates and RMPs of the child nodes $\{\mathtt{v}_j\}_{j=1}^N$ define, for the parent node $\mathtt{u}$, a Lyapunov function candidate

$$V_{\mathtt{u}}(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{2}\, \mathbf{x}^\top\, \mathbf{G}_{\mathtt{u}}(\mathbf{x}, \dot{\mathbf{x}})\, \dot{\mathbf{x}} + \Phi_{\mathtt{u}}(\mathbf{x}). \tag{13.11}$$

where $\mathbf{G}_{\mathtt{u}}$ and $\Phi_{\mathtt{u}}$ are given in (13.4).

The following lemma states how the decay-rate of $V_{\mathtt{u}}$ is connected to the decay-rates of $\{V_{\mathtt{v}_j}\}_{j=1}^N$ via `pullback`.

Lemma 13.3.1. *For each child node* $\mathtt{v}_j$*, assume that* $\mathbf{f}_{\mathtt{v}_j} = \mathbf{M}_{\mathtt{v}_j}\, \ddot{\mathbf{y}}_j$ *renders* $\dot{V}_{\mathtt{v}_j}(\mathbf{y}_j, \dot{\mathbf{y}}_j) = -U_{\mathtt{v}_j}(\mathbf{y}_j, \dot{\mathbf{y}}_j)$ *for* $V_{\mathtt{v}_j}$ *in (13.10). If the parent node* $\mathtt{u}$ *follows dynamics* $\mathbf{f}_{\mathtt{u}} = \mathbf{M}_{\mathtt{u}}(\mathbf{x}, \dot{\mathbf{x}})\, \ddot{\mathbf{x}}$*, where* $\mathbf{f}_{\mathtt{u}}$ *and* $\mathbf{M}_{\mathtt{u}}$ *are given by* `pullback`*, then* $\dot{V}_{\mathtt{u}}(\mathbf{x}, \dot{\mathbf{x}}) = -\sum_{j=1}^N U_{\mathtt{v}_j}(\mathbf{y}_j, \dot{\mathbf{y}}_j)$ *for* $V_{\mathtt{u}}$ *in (13.11).*

Proof. For notational convenience, we suppress the arguments of functions. First, note that,

$$V_{\mathtt{u}} = \frac{1}{2} \sum_{j=1}^N \mathbf{x}^\top\, \mathbf{J}_{\mathtt{e}_j}^\top\, \mathbf{G}_{\mathtt{v}_j}\, \mathbf{J}_{\mathtt{e}_j}\, \dot{\mathbf{x}} + \sum_{j=1}^N \Phi_{\mathtt{v}_j} = \sum_{j=1}^N V_{\mathtt{v}_j}. \tag{13.12}$$

As $\mathbf{G}_{\mathtt{v}_j}$ is a function in both $\mathbf{y}$ and $\dot{\mathbf{y}}$, following a similar derivation as in (Cheng et al.,

2018b), we can show

$$\dot{V}_{\mathtt{u}} = \sum_{j=1}^{N} \dot{\mathbf{y}}_j^\top \mathbf{M}_{\mathtt{v}_j} \ddot{\mathbf{y}}_j + \frac{1}{2} \dot{\mathbf{y}}_j^\top \overset{\mathbf{y}_j}{\mathbf{G}}_{\mathtt{v}_j} \dot{\mathbf{y}}_j + \dot{\mathbf{y}}_j^\top \nabla_{\mathbf{y}_j} \Phi_{\mathtt{v}_j}, \tag{13.13}$$

where $\mathbf{M}_{\mathtt{v}_j}$ and $\overset{\mathbf{y}_j}{\mathbf{G}}_{\mathtt{v}_j}$ are the inertial matrix and the curvature term defined in Section 13.2.1.

Note that $\ddot{\mathbf{y}}_j = \mathbf{J}_{\mathtt{e}_j} \ddot{\mathbf{x}} + \dot{\mathbf{J}}_{\mathtt{e}_j} \dot{\mathbf{x}}$, where $\ddot{\mathbf{x}}$ is given by the RMP $[\mathbf{f}_{\mathtt{u}}, \mathbf{M}_{\mathtt{u}}]^{\mathcal{M}}$. Hence, the first term can be rewritten as

$$\begin{aligned}
&\sum_{j=1}^{N} \dot{\mathbf{y}}_j^\top \mathbf{M}_{\mathtt{v}_j} \ddot{\mathbf{y}}_j = \dot{\mathbf{x}}^\top \left(\sum_{j=1}^{N} \mathbf{J}_{\mathtt{e}_j}^\top \mathbf{M}_{\mathtt{v}_j} \left(\mathbf{J}_{\mathtt{e}_j} \ddot{\mathbf{x}} + \dot{\mathbf{J}}_{\mathtt{e}_j} \dot{\mathbf{x}} \right) \right) \\
&= \dot{\mathbf{x}}^\top \mathbf{f}_{\mathtt{u}} + \dot{\mathbf{x}}^\top \left(\sum_{j=1}^{N} \mathbf{J}_{\mathtt{e}_j}^\top \mathbf{M}_{\mathtt{v}_j} \dot{\mathbf{J}}_{\mathtt{e}_j} \dot{\mathbf{x}} \right) \\
&= \dot{\mathbf{x}}^\top \left(\sum_{j=1}^{N} \mathbf{J}_{\mathtt{e}_j}^\top \left(\mathbf{f}_{\mathtt{v}_j} - \mathbf{M}_{\mathtt{v}_j} \dot{\mathbf{J}}_{\mathtt{e}_j} \dot{\mathbf{x}} \right) \right) + \dot{\mathbf{x}}^\top \left(\sum_{j=1}^{N} \mathbf{J}_{\mathtt{e}_j}^\top \mathbf{M}_{\mathtt{v}_j} \dot{\mathbf{J}}_{\mathtt{e}_j} \dot{\mathbf{x}} \right) \\
&= \dot{\mathbf{x}}^\top \left(\sum_{j=1}^{N} \mathbf{J}_{\mathtt{e}_j}^\top \mathbf{f}_{\mathtt{v}_j} \right) = \sum_{j=1}^{N} \dot{\mathbf{y}}_j^\top \mathbf{f}_{\mathtt{v}_j}.
\end{aligned}$$

The time-derivative of $V_{\mathtt{u}}$ can then be simplified as

$$\dot{V}_{\mathtt{u}} = \sum_{j=1}^{N} \dot{\mathbf{y}}_j^\top \mathbf{f}_{\mathtt{v}_j} + \frac{1}{2} \dot{\mathbf{y}}_j^\top \overset{\mathbf{y}_j}{\mathbf{G}}_{\mathtt{v}_j} \dot{\mathbf{y}}_j + \dot{\mathbf{y}}_j^\top \nabla_{\mathbf{y}_j} \Phi_{\mathtt{v}_j}. \tag{13.14}$$

By assumption on $\mathtt{v}_j$, we also have

$$\dot{\mathbf{y}}_j^\top \mathbf{f}_{\mathtt{v}_j} + \frac{1}{2} \dot{\mathbf{y}}_j^\top \overset{\mathbf{y}_j}{\mathbf{G}}_{\mathtt{v}_j} \dot{\mathbf{y}}_j + \dot{\mathbf{y}}_j^\top \nabla_{\mathbf{y}_j} \Phi_{\mathtt{v}_j} = -U_{\mathtt{v}_j}(\mathbf{y}_j, \dot{\mathbf{y}}_j). \tag{13.15}$$

The statement follows then from the two equalities. ■

We can use Lemma 13.3.1 to infer the overall stability of RMPflow. For an RMP-tree with K leaf nodes, let leaf node $\mathtt{l}_k$ be defined on task space $\mathcal{T}_k$ with coordinates $\mathbf{z}_k$ and has a natural Lyapunov function candidate

$$V_{\mathtt{l}_k}(\mathbf{z}_k, \dot{\mathbf{z}}_k) = \frac{1}{2} \dot{\mathbf{z}}_k^\top \mathbf{G}_{\mathtt{l}_k}(\mathbf{z}_k, \dot{\mathbf{z}}_k) \dot{\mathbf{z}}_k + \Phi_{\mathtt{l}_k}(\mathbf{z}_k). \tag{13.16}$$

for some potential function $\Phi_{\mathtt{l}_k}$ and positive-definite Riemannian metric $\mathbf{G}_{\mathtt{l}_k}$ defined as

above. By Lemma 13.3.1, if each leaf node $\mathtt{l}_k$ satisfies a CLF constraint,

$$\dot{V}_{\mathtt{l}_k}(\mathbf{z}_k, \dot{\mathbf{z}}_k) = -U_{\mathtt{l}_k}(\mathbf{z}_k, \dot{\mathbf{z}}_k), \tag{13.17}$$

then a similar constraint is satisfied by the root node. This observation is summarized below without proof.

Proposition 13.3.1. *For each leaf node $\mathtt{l}_k$, assume that $\mathbf{f}_{\mathtt{l}_k} = \mathbf{M}_{\mathtt{l}_k}\ddot{\mathbf{z}}_k$ renders (13.17) for (13.16). Consider the Lyapunov function candidate at the root node $V_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}})$ defined through (13.11). Then, for the root node control policy of RMPflow, it holds $\dot{V}_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}}) = -\sum_{k=1}^{K} U_{\mathtt{l}_k}(\psi_{\mathtt{r}\to\mathtt{l}_k}(\mathbf{q}), \mathbf{J}_{\mathtt{r}\to\mathtt{l}_k}\dot{\mathbf{q}})$, where $\psi_{\mathtt{r}\to\mathtt{l}_k}$ is the map from $\mathcal{C}$ to $\mathcal{T}_k$, which can be obtained through composing maps from the root node $\mathtt{r}$ to the leaf node $\mathtt{l}_k$ on the RMP-tree, and $\mathbf{J}_{\mathtt{r}\to\mathtt{l}_k} = \partial_{\mathbf{q}}\psi_{\mathtt{r}\to\mathtt{l}_k}$.*

Note that Proposition 13.3.1 provides an alternative way to show the stability results of RMPflow.

Corollary 13.3.1. *For each leaf node $\mathtt{l}_k$, assume that $\mathbf{f}_{\mathtt{l}_k}$ is given by a GDS $(\mathcal{T}_k, \mathbf{G}_{\mathtt{l}_k}, \mathbf{B}_{\mathtt{l}_k}, \Phi_{\mathtt{l}_k})$. Consider the Lyapunov function candidate at the root node $V_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}})$ defined recursively through (13.11). Then we have, $\dot{V}_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}}) = -\dot{\mathbf{q}}^\top \mathbf{B}_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ under the resulting control policies from RMPflow, where $\mathbf{B}_{\mathtt{r}}$ is defined recursively through (13.4).*

With Corollary 13.3.1, we then can show Theorem 13.2.1 by invoking LaSalle's invariance principle (Khalil, 1996).

13.3.2 Global Stability Properties

More importantly, by Proposition 13.3.1, we can find how CLF constraints are propagated from the leaf nodes to the root node through `pullback`.

Proposition 13.3.2. *For each leaf node $\mathtt{l}_k$, assume that $\mathbf{f}_{\mathtt{l}_k} = \mathbf{M}_{\mathtt{l}_k}\ddot{\mathbf{z}}_{\mathtt{l}_k}$ renders, for $V_{\mathtt{l}_k}$ in* (13.16)*,*

$$\dot{V}_{\mathtt{l}_k}(\mathbf{z}_k, \dot{\mathbf{z}}_k) \leq -\alpha_k(\|\dot{\mathbf{z}}_k\|), \tag{13.18}$$

where α_k is a locally Lipschitz continuous class $\mathcal{K}$ functions (Khalil, 1996). Consider the Lyapunov function candidate at the root node $V_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}})$ defined recursively through (13.11). Then

$$\dot{V}_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}}) \leq -\sum_{k=1}^{K} \alpha_k \left(\| \mathbf{J}_{\mathtt{r} \to \mathtt{l}_k} \dot{\mathbf{q}} \| \right) \tag{13.19}$$

under the resulting control policies from RMPflow.

With this insight, we state a new stability theorem of RMPflow by applying LaSalle's invariance principle. We assume that the inertia matrix at the root node $\mathbf{M}_{\mathtt{r}}$ is nonsingular for simplicity, so that the actual control input can be solved through the `resolve` operation; a sufficient condition for $\mathbf{M}_{\mathtt{r}}$ being nonsingular is provided in Chapter 11.

Theorem 13.3.1. *For each leaf node $\mathtt{l}_k$, assume that $\mathbf{f}_{\mathtt{l}_k} = \mathbf{M}_{\mathtt{l}_k} \ddot{\mathbf{z}}_{\mathtt{l}_k}$ renders* (13.18)*. Suppose that $\mathbf{M}_{\mathtt{r}}$ is nonsingular, and the task space $\mathcal{T}$ is an immersion of the configuration space $\mathcal{C}$. Then the control policy generated by RMPflow renders the system converging to the forward invariant set*

$$\mathcal{C}_\infty \coloneqq \left\{ (\mathbf{q}, \dot{\mathbf{q}}) : \dot{\mathbf{q}} = 0, \ \sum_{j=1}^{K} \mathbf{J}_{\mathtt{r} \to \mathtt{l}_k}^\top \mathbf{f}_{\mathtt{l}_k} = 0 \right\}. \tag{13.20}$$

Further if, for all leaf nodes, $\mathbf{f}_{\mathtt{l}_k} = -\nabla_{\mathbf{z}_k} \Phi_{\mathtt{l}_k}(\mathbf{z}_k)$ when $\dot{\mathbf{z}}_k = 0$, the system converges to the forward invariant set

$$\mathcal{C}_\infty^\Phi \coloneqq \{ (\mathbf{q}, \dot{\mathbf{q}}) : \nabla_{\mathbf{q}} \Phi_{\mathtt{r}}(\mathbf{q}) = 0, \dot{\mathbf{q}} = 0 \}, \tag{13.21}$$

where $\Phi_{\mathtt{r}}$ is the potential in $V_{\mathtt{r}}$ defined recursively in (13.4).

Proof. By assumption, $V_{\mathtt{r}}$ is proper, continuously differentiable and lower bounded. Hence, the system converges to the largest invariant set in $\{ (\mathbf{q}, \dot{\mathbf{q}}) : \dot{V}_{\mathtt{r}}(\mathbf{q}, \dot{\mathbf{q}}) = 0 \}$ by LaSalle's invariance principle (Khalil, 1996). By (13.19) in Proposition 13.3.2, $\dot{V}_{\mathtt{r}} = 0$ if and only if $\mathbf{J}_{\mathtt{r} \to \mathtt{l}_k} \dot{\mathbf{q}} = 0$ for all $k = 1, \ldots, K$. Since $\mathcal{C}$ is immersed in $\mathcal{T}$, we have $\dot{\mathbf{q}} = 0$. Hence, the

system converges to a forward invariant set $\mathcal{C}_\infty \coloneqq \{(\mathbf{q}, \dot{\mathbf{q}}) : \dot{\mathbf{q}} = 0\}$. Any forward invariant set in $\mathcal{C}_\infty$ must have $\ddot{\mathbf{q}} = 0$, which implies that $\mathbf{f}_\mathtt{r} = 0$ as $\mathbf{M}_\mathtt{r}$ is nonsingular. Note that $\mathbf{f}_\mathtt{r}$ is given by the `pullback` operation recursively, hence,

$$0 = \mathbf{f}_\mathtt{r} = \sum_{k=1}^{K} \mathbf{J}_{\mathtt{r}\to\mathtt{l}_k}^\top \left(\mathbf{f}_{\mathtt{l}_k} - \mathbf{M}_{\mathtt{l}_k} \dot{\mathbf{J}}_{\mathtt{r}\to\mathtt{l}_k} \dot{\mathbf{q}}\right) = \sum_{k=1}^{K} \mathbf{J}_{\mathtt{r}\to\mathtt{l}_k}^\top \mathbf{f}_{\mathtt{l}_k}$$

where the last equality follows from $\dot{\mathbf{q}} = 0$. Thus, the system converges to the forward invariant set in (13.20).

Now, assume that $\mathbf{f}_{\mathtt{l}_k} = -\nabla_{\mathbf{z}_k} \Phi_{\mathtt{l}_k}(\mathbf{z}_k)$ when $\dot{\mathbf{z}}_k = 0$ (which is implied by $\dot{\mathbf{q}} = 0$). Notice that by the definition of $\Phi_\mathtt{r}$ in (13.4), $\Phi_\mathtt{r}(\mathbf{q}) = \sum_{k=1}^{K} \Phi_{\mathtt{l}_k}(\mathbf{z}_k)$. By the chain rule,

$$\nabla_\mathbf{q} \Phi_\mathtt{r}(\mathbf{q}) = \sum_{k=1}^{K} \mathbf{J}_{\mathtt{r}\to\mathtt{l}_k}^\top \nabla_{\mathbf{z}_k} \Phi_{\mathtt{l}_k}(\mathbf{z}_k) = -\sum_{k=1}^{K} \mathbf{J}_{\mathtt{r}\to\mathtt{l}_k}^\top \mathbf{f}_{\mathtt{l}_k}.$$

Hence $\sum_{k=1}^{K} \mathbf{J}_{\mathtt{r}\to\mathtt{l}_k}^\top \mathbf{f}_{\mathtt{l}_k} = 0$ implies $\nabla_\mathbf{q} \Phi_\mathtt{r}(\mathbf{q}) = 0$. Thus, the system converges forwardly to (13.21). ■

Theorem 13.3.1 implies that subtask controllers satisfying CLF constraints (13.18) can be stably combined by RMPflow.

13.4 A Computational Framework for RMPflow with CLF Constraints

We introduce a computational framework for controller synthesis based on the stability results presented in Section 13.3. The main idea is to leverage Proposition 13.3.2, which says that RMPflow is capable of preserving CLF constraints in certain form. Recall that for leaf node $\mathtt{v}_j$, the constraint on the time-derivative of the Lyapunov function in Proposition 13.3.2 is $\dot{V}_{\mathtt{l}_k}(\mathbf{z}_k, \dot{\mathbf{z}}_k) \leq -\alpha_k(\|\dot{\mathbf{z}}_k\|)$. Combined with the particular choice of leaf-node Lyapunov function candidate in (13.16), this yields a CLF constraint[3]

$$\dot{\mathbf{z}}_k^\top \mathbf{f}_{\mathtt{l}_k} \leq -\dot{\mathbf{z}}_k^\top \left(\nabla_{\mathbf{z}_k} \Phi_{\mathtt{l}_k} + \boldsymbol{\xi}_{\mathbf{G}_{\mathtt{l}_k}}\right) - \alpha_k(\|\dot{\mathbf{z}}_k\|), \tag{13.22}$$

[3]This a linear constraint with respect to $\mathbf{f}_{\mathtt{l}_k}$. When $\dot{\mathbf{z}}_k = 0$, the constraint (13.22) holds trivially because both sides equal 0.

where $\boldsymbol{\xi}_{\mathbf{G}_{1_k}}$ is defined in (13.1). Proposition 13.3.2 shows that, when the leaf-node control policies satisfy (13.22), RMPflow will yield a stable controller under suitable conditions. This provides a constructive principle to synthesize controllers.

13.4.1 Algorithm Details

Assume that some nominal controller $\mathbf{u}^d_{1_k}$ is provided by the specification of subtask k. We design the leaf-node controller as a *minimally invasive* controller that modifies the nominal controller as little as possible while satisfying the CLF constraint (13.22):

$$\begin{aligned} \mathbf{f}^*_{1_k} \quad &= \quad \arg\min_{\mathbf{f}_{1_k}} \|\mathbf{f}_{1_k} - \mathbf{M}_{1_k}\,\mathbf{u}^d_{1_k}\|^2_{\mathbf{P}_{1_k}} \\ &\text{s.t.} \quad \dot{\mathbf{z}}_k^\top\,\mathbf{f}_{1_k} \;\leq\; -\dot{\mathbf{z}}_k^\top\left(\nabla_{\mathbf{z}_k}\Phi_{1_k} + \boldsymbol{\xi}_{\mathbf{G}_{1_k}}\right) - \alpha_k(\|\dot{\mathbf{z}}_k\|) \end{aligned} \tag{13.23}$$

where $\mathbf{P}_{1_k} \succ 0$ and $\mathbf{M}_{1_k}$ is given by $\mathbf{G}_{1_k}$ through (13.2). Possible choices of $\mathbf{P}_{1_k}$ include the identity matrix $\mathbf{I}$ and the inverse of the inertial matrix $\mathbf{M}^{-1}_{1_k}$. In particular, $\mathbf{P}_{1_k} = \mathbf{M}^{-1}_{1_k}$ yields an objective function equivalent to $\|\mathbf{a}_{1_k} - \mathbf{u}^d_{1_k}\|^2_{\mathbf{M}_{1_k}}$, where $\mathbf{a}_{1_k}$ is the acceleration policy of the node.

We combine this minimally invasive controller design with RMPflow as a new computational framework for controller synthesis, called *RMPflow–CLF*. RMPflow–CLF follows the same procedure as the original RMPflow in Chapter 11 as is discussed in Section 13.2.1. The difference is that the leaf nodes solve for the RMPs based on the optimization problem (13.23) during the evaluation step. Note that (13.23) is a QP problem with a single linear constraint, so it can be solved analytically by projecting $\mathbf{M}_{1_k}\mathbf{u}^d_{1_k}$ onto the half-plane given by the constraint.

13.4.2 Stability Properties

The form of (13.23) together with Theorem 13.3.1 and the results of (Morris, Powell, and Ames, 2013) yields the following theorem:

Theorem 13.4.1. *Under the assumptions in Theorem 13.3.1, if* $\{\mathbf{u}^d_{\mathtt{1}_k}\}_{k=1}^K$, $\{\mathbf{M}_{\mathtt{1}_k}\}_{k=1}^K$, *all edge Jacobians and their derivatives are locally Lipschitz continuous, then the control policy generated by RMPflow–CLF is locally Lipschitz continuous and renders the system converging forwardly to* (13.20).

Proof. By Theorem 13.3.1, the system converges to (13.20). By (Morris, Powell, and Ames, 2013), for all $k \in \{1, \ldots, K\}$, $\mathbf{f}_{\mathtt{1}_k}$ is locally Lipschitz. Since under the assumption `pullback` and `resolve` preserves Lipschitz continuity; the statement follows. ■

Note that RMPflow can be interpreted as a soft version of the QP–CLF formulation (Morris, Powell, and Ames, 2013) that enforces the decay-rates of *all* Lyapunov function candidates (13.8). Meanwhile, compared with the QP–CLF framework with slack variables (Ames, Grizzle, and Tabuada, 2014) that requires the users to design the objective function trade off between control specifications, RMPflow provides a structured way to implicitly generate such an objective function so that the system is *always* stable.

It should be noted that the system can also be stabilized by directly enforcing a single constraint on the time derivative of the combined Lyapunov function candidate (13.19) at the root node, rather than enforcing the CLF constraint at every leaf node (13.18). However, this can be less desirable: although the stability can be guaranteed for the resulting controller, the behavior of each individual subtask is no longer explicitly regulated. By contrast, the approach with leaf-node CLF constraints allows the users to design and test the controllers from (13.23) independently. This allows for designing controllers that can be applied to robots with different kinematic structures, which is a significant feature of RMPflow (Cheng et al., 2018b).

13.5 Experimental Results

In this section, we compare the proposed RMPflow–CLF framework with the original RMPflow framework in Chapter 11 (Cheng et al., 2018b). A video of the experimental

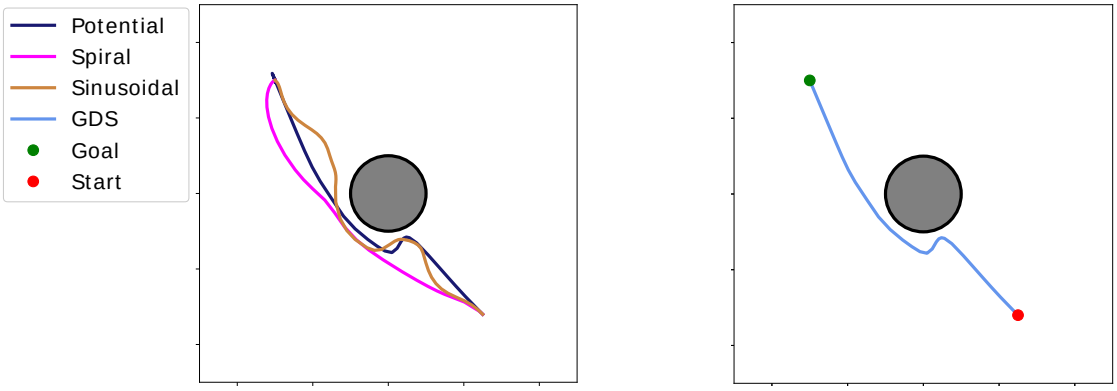
results can be found at `https://youtu.be/eU_x8Yklv-4`. The original RMPflow framework (Cheng et al., 2018b) is referred to as RMPflow–GDS to differentiate it from RMPflow–CLF.

13.5.1 Simulation Results

We present two simulated examples, a 2-dimensional goal reaching task and a multi-robot goal reaching example.

2D Goal Reaching

We first consider the 2D goal reaching task presented in Chapter 11 (Cheng et al., 2018b). In this example, a planar robot with double-integrator dynamics is expected to move to a goal position without colliding into a circular obstacle. As is in (Cheng et al., 2018b), the RMP-tree has a collision avoidance leaf-node RMP and a goal attractor leaf-node RMP. For the RMPflow–CLF framework, we use the collision avoidance RMP in (Cheng et al., 2018b) and keep the choice of metrics and potential functions for the goal attractor RMP consistent with (Cheng et al., 2018b). For the goal attractor RMP, we present several nominal controllers: (i) a pure potential-based nominal controller $\mathbf{f}_{pt}^d = \mathbf{M}\mathbf{u}_{pt}^d = -\nabla\Phi$; (ii) a spiral nominal controller $\mathbf{f}_{sp}^d = -\nabla\Phi + \|\dot{\mathbf{z}}\|\,\mathbf{v}$, where $\mathbf{v}$ is the potential-based controller rotated by $\pi/2$, i.e. $\mathbf{v} = -R(\pi/2)\,\nabla\Phi$ with $R(\cdot)$ being the rotation matrix; and (iii) a sinusoidal controller $\mathbf{f}_{sn}^d = -\nabla\Phi + \sin(t/4)\,\|\dot{\mathbf{z}}\|\,\mathbf{v}$. For the minimally invasive controller, we use $\mathbf{P} = \mathbf{I}$ to minimize the Euclidean distance between the nominal controller and the solution to the optimization problem (13.23). We implement the RMPflow–GDS framework with the same choice of parameters as (Cheng et al., 2018b). The trajectories under different nominal controllers are shown in Fig. 13.2. Although it is possible that similar behaviors can be realized with the RMPflow–GDS framework with a careful redesign of the metric and potential function, the RMPflow–CLF framework can produce a rich class of behaviors without being concerned with the geometric properties of the subtask manifold.



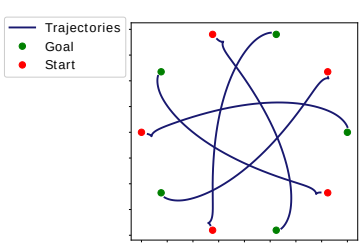
(a) RMPflow–CLF (b) RMPflow–GDS

Figure 13.2: 2D goal reaching task with a circular obstacle (grey). (a) RMPflow–CLF with three choices of nominal controllers, resulting in different goal reaching behaviors. (b) RMPflow–GDS with the goal attractor given by a GDS. The behavior is limited by the choice of the metric and the potential function.

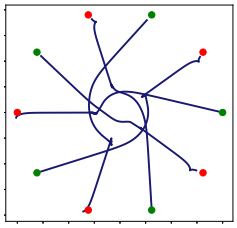
Multi-Robot Goal Reaching

RMPflow–CLF guarantees system stability even when the nominal controllers are not inherently stable or asymptotically stable. Therefore, the user can incorporate design knowledge given by, e.g. motion planners, human demonstrations or even heuristics, into the nominal controllers. To illustrate this, we consider a multi-robot goal reaching task, where the robots are tasked with moving to the opposite side of the arena without colliding. If the robots move in straight lines, their trajectories would intersect near the center of the arena. Due to the symmetric configuration, the system can easily deadlock with robots moving very slowly or stopping near the center to avoid collisions. This problem can be fixed if the symmetry is broken. One possible solution is to design nominal controllers for the goal attractors so that the robots move along curves.

We compare the spiral goal attractor RMP with the GDS goal attractor RMP presented in (Li et al., 2019a). In both cases, an RMP-tree structure similar to the centralized RMP-tree structure in (Li et al., 2019a) is used. We define collision avoidance for pairs of robots in the RMP-tree with the same choice of parameters. The trajectories of the robots under the spiral nominal controllers are shown in Fig. 13.3a. The spiral controllers produce smooth



(a) RMPflow–CLF



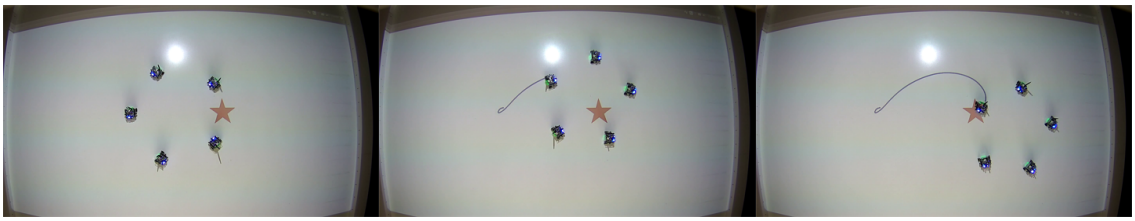
(b) RMPflow–GDS (Cheng et al., 2018b)

Figure 13.3: Multi-robot goal reaching task. (a) RMPflow–CLF with spiral nominal controllers. The robots move to their goal smoothly. (b) RMPflow–GDS with the goal attractor given by a GDS. Due to the symmetry of the configuration, the system suffers from deadlock when the robots are near the center: the robots oscillate around the deadlock configuration.

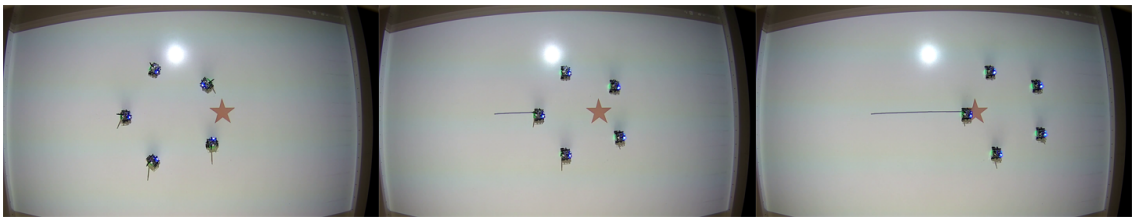
motion, whereas the GDS goal attractors produce jerky motion when the robots are near the center due to deadlock caused by the symmetric configuration (Fig. 13.3b).

13.5.2 Robotic Implementation

We present an experiment conducted on the Robotarium (Pickem et al., 2017), a remotely accessible swarm robotics platform. In the experiment, five robots are tasked with preserving a regular pentagon formation while the leader has an additional task of reaching a goal. We use the same RMP-tree structure and parameters for most leaf-node RMPs as described in the formation preservation experiment in (Li et al., 2019a). The only difference is that we replace the GDS goal attractor in (Li et al., 2019a) with the spiral nominal controller augmented with the CLF condition (13.23). Fig. 13.4 presents the snapshots from the formation preservation experiment. In Fig. 13.4a–Fig. 13.4c, we see that the leader approaches the goal with a spiral trajectory specified by the nominal controller, while other subtask controllers preserve distances and avoid collision. This shows the efficacy of our controller synthesis framework. By contrast, the robot moves in straight lines under the goal attractor given by the GDS (see Fig. 13.4d-Fig. 13.4f). Although it could be possible to redesign the subtask manifold such that there exists a GDS that produces similar behaviors, the RMPflow–CLF framework provides the user additional flexibility to shape the



(a) Spiral CLF controller: $t = 0s$ (b) Spiral CLF controller: $t = 18s$ (c) Spiral CLF controller: $t = 38s$



(d) GDS controller: $t = 0s$ (e) GDS controller: $t = 8s$ (f) GDS controller: $t = 19s$

Figure 13.4: Multi-robot formation preservation task. The robots are tasked with preserving a regular pentagon formation while the leader has an additional task of reaching a goal position. (a) RMPflow–CLF with a spiral nominal controller. (b) RMPflow–GDS. The goal (red star) and the trajectories (blue curves) of the leader robot are projected onto the environment through an overhead projector. RMPflow–CLF shapes the goal-reaching behavior through a spiral nominal controller.

behaviors without worrying about the geometric properties of the subtask manifolds.

13.6 Conclusions

We consider robot control with multiple control specifications by adopting Riemannian Motion Policies (RMPs), a recent concept in robotics for describing control policies on manifolds, and RMPflow, the computational structure to combine these controllers. The stability results of RMPflow is re-established and extended through a rigorous CLF treatment. This new analysis suggests that any subtask controllers satisfying certain CLF constraints can be stably combined by RMPflow, while the original analysis given in (Cheng et al., 2018b) only provides stability guarantees for a limited type of controller. Based on this analysis, we propose a new computational framework, RMPflow–CLF, to stably combine individually designed subtask controllers. This formulation provides users the flexibility of shaping behaviors of subtasks through nominal subtask controllers given by, e.g. heuristics,

human demonstrations, and motion planners. The proposed RMPflow–CLF framework is validated through numerical simulation and deployment on real robots.

CHAPTER 14

EPILOGUE

We have demonstrated through the previous chapters how better theoretical formulations of fundamental problems can address practical needs and impact the direction of algorithm design. From Chapter 3 to 5, we illustrate how imitation learning (IL) can be used as an interface to take advantage of prior knowledge in speeding up policy optimization. Driven by these exciting results, we refine theoretical analyses of IL in Chapter 6 and 7. These new insights then lead to the development of new online learning theory and algorithms (from Chapter 8 to 10) that advance the state-of-the-art techniques for a large set of learning problems faced in practice, including reinforcement learning (RL). In parallel, we present a foundational framework for policy fusion and structural policy design, called RMPflow, in Chapter 11. The development of RMPflow is motivated by the need of computationally efficient, stable controllers that are capable of generating flexible behaviors and working consistently across various robots. Rather than tweaking existing designs, we deeply investigate the source of deficiency in current designs and build a new theoretical framework bottom-up. Consequently, RMPflow can systematically address these practical needs as we showcase in Chapter 11 and 13. We also show in Chapter 12 that RMPflow can be used as a stable policy class in policy optimization, so that the statistical learning techniques developed in this thesis can be safely applied to robots.

Besides policy optimization and structural policy fusion, we have also used the same principle to design structured kernels for learning robot dynamics (Cheng and Huang, 2016; Cheng et al., 2016), linear-time algorithms for learning large-scale Gaussian process models (Cheng and Boots, 2016, 2017; Salimbeni et al., 2018), a fast bi-level optimization algorithm using truncating back-propagation (Shaban et al., 2019), a family of MPC algorithms based on dynamic mirror descent (Wagener et al., 2019), and trajectory-wise control

variates for variance reduction in policy gradient methods (Cheng, Yan, and Boots, 2019). All these research outcomes highlight the significance of having a good theoretical perspective in designing practical algorithms.

Maintaining a reciprocal relationship between theory and practice thus is a key to closing the reality gap. I believe that theoreticians can play an important role in this endeavor: the abstract way of mathematical thinking is invaluable to understanding the underlying phenomena in real-world problems, asking the right research questions, and then providing systematic, constructive solutions. While my research so far has successfully taken steps toward a more principled approach to efficient learning and algorithm design, it also opens the door to many other interesting research directions. For example, systematically balancing sample and computation budgets in real-world policy optimization is an important topic, and determining how to do this correctly would naturally unify on-policy and off-policy learning. The PICCOLO framework in Chapter 10, which combines model-free and model-based learning by decoupling sample and optimization complexities, is a preliminary attempt at achieving this goal. On the other hand, parallelizing the computation of RMPflow can have significant practical impacts, especially in building larger-scale systems, such as humanoids and multi-agent systems. The identification between RMPflow and sparse linear solvers in Chapter 11 can be an important step to solving this problem.

I am interested in building complex sequential-decision-making agents that can learn to adapt and work robustly in real-world situations with minimal engineering and human intervention. Solving this long-standing problem inevitably requires a holistic integration of techniques developed in various disciplines. To amortize this explosion of design complexity, new theories that thoroughly outline their interactions are imperative. I hope that, at this very end, you are also convinced of the importance of developing theories alongside real-world applications. May the research here encourage future endeavors toward this direction. Congratulations on finishing reading the thesis!

REFERENCES

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. (2016). "Tensorflow: a system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation*. Vol. 16, pp. 265–283.

Abbeel, Pieter and Andrew Y Ng (2005). "Exploration and apprenticeship learning in reinforcement learning". In: *International Conference on Machine learning*. ACM, pp. 1–8.

Abernethy, Jacob, Peter L Bartlett, and Elad Hazan (2011). "Blackwell approachability and no-regret learning are equivalent". In: *Annual Conference on Learning Theory*, pp. 27–46.

Albu-Schaffer, Alin and Gerd Hirzinger (2002). "Cartesian impedance control techniques for torque controlled light-weight robots". In: *IEEE International Conference on Robotics and Automation*. Vol. 1, pp. 657–663.

Alexander, Stephanie, Richard Bishop, and Robert Ghrist (2006). "Pursuit and Evasion in Non-convex Domains of Arbitrary Dimensions". In: *Robotics: Science and Systems*, pp. 277–310.

Amari, Shun-Ichi (1998). "Natural gradient works efficiently in learning". In: *Neural Computation* 10.2, pp. 251–276.

Ames, Aaron D, Jessy W Grizzle, and Paulo Tabuada (2014). "Control barrier function based quadratic programs with application to adaptive cruise control". In: *IEEE Conference on Decision and Control*. IEEE, pp. 6271–6278.

Amos, Brandon, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter (2018). "Differentiable MPC for End-to-end Planning and Control". In: *Advances in Neural Information Processing Systems*, pp. 8299–8310.

Anthony, Thomas, Zheng Tian, and David Barber (2017). "Thinking fast and slow with deep learning and tree search". In: *Advances in Neural Information Processing Systems*, pp. 5360–5370.

Antsaklis, Panos J and Anthony N Michel (2007). *A linear systems primer*. Vol. 1. Birkhäuser Boston.

Argall, Brenna D, Sonia Chernova, Manuela Veloso, and Brett Browning (2009). "A survey of robot learning from demonstration". In: *Robotics and Autonomous Systems* 57.5, pp. 469–483.

Arkin, Ronald C (2008). "Governing lethal behavior: embedding ethics in a hybrid deliberative/reactive robot architecture". In: *ACM/IEEE International Conference on Human Robot Interaction*. ACM, pp. 121–128.

Arkin, Ronald C et al. (1998). *Behavior-based robotics*. MIT press.

Beck, Amir and Marc Teboulle (2003). "Mirror descent and nonlinear projected subgradient methods for convex optimization". In: *Operations Research Letters* 31.3, pp. 167–175.

Bellman, Richard (1954). "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6, pp. 503–515.

Bertsekas, Dimitri P, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas (1995). *Dynamic programming and optimal control*. Vol. 1. 2. Athena scientific Belmont, MA.

Besbes, Omar, Yonatan Gur, and Assaf Zeevi (2015). "Non-stationary stochastic optimization". In: *Operations research* 63.5, pp. 1227–1244.

Bianchi, M and S Schaible (1996). "Generalized monotone bifunctions and equilibrium problems". In: *Journal of Optimization Theory and Applications* 90.1, pp. 31–43.

Blum, Eugen (1994). "From optimization and variational inequalities to equilibrium problems". In: *Math. student* 63, pp. 123–145.

Bojarski, Mariusz, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, and Jiakai Zhang (2016). "End to end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316*.

Bojarski, Mariusz, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller (2017). "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car". In: *arXiv preprint arXiv:1704.07911*.

Borrelli, Francesco, Paolo Falcone, Tamas Keviczky, Jahan Asgari, and Davor Hrovat (2005). "MPC-based approach to active steering for autonomous vehicle systems". In: *International Journal of Vehicle Autonomous Systems* 3.2, pp. 265–291.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex optimization*. Cambridge university press.

Bregman, Lev M (1967). "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming". In: *USSR Computational Mathematics and Mathematical Physics* 7.3, pp. 200–217.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). "OpenAI Gym". In: *arXiv preprint arXiv:1606.01540*.

Bullo, Francesco and Andrew D Lewis (2004). *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Vol. 49. Springer Science & Business Media.

Burachik, Regina S and R Díaz Millán (2019). "A projection algorithm for non-monotone variational inequalities". In: *Set-Valued and Variational Analysis*.

Cesa-Bianchi, Nicolo, Alex Conconi, and Claudio Gentile (2004). "On the generalization ability of on-line learning algorithms". In: *IEEE Transactions on Information Theory* 50.9, pp. 2050–2057.

Cesa-Bianchi, Nicolo and Gabor Lugosi (2006). *Prediction, learning, and games*. Cambridge university press.

Chang, Kai-Wei, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford (2015). "Learning to search better than your teacher". In: *International Conference on Machine Learning*.

Chebotar, Yevgen, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine (2017). "Combining model-based and model-free updates for trajectory-centric reinforcement learning". In: *International Conference on Machine Learning-Volume 70*, pp. 703–711.

Chen, Yichen, Lihong Li, and Mengdi Wang (2018). "Scalable Bilinear π Learning Using State and Action Features". In: *arXiv preprint arXiv:1804.10328*.

Chen, Yichen and Mengdi Wang (2016). "Stochastic primal-dual methods and sample complexity of reinforcement learning". In: *arXiv preprint arXiv:1612.02516*.

Cheng, Ching-An (2018). "Policy Optimization as Predictable Online Learning Problems: Imitation Learning and Beyond". Microsoft Research Talks.

Cheng, Ching-An and Byron Boots (2016). "Incremental Variational Sparse Gaussian Process Regression". In: *Advances in Neural Information Processing Systems*, pp. 4410–4418.

— (2017). "Variational Inference for Gaussian Process Models with Linear Complexity". In: *Advances in Neural Information Processing Systems*, pp. 5184–5194.

Cheng, Ching-An and Byron Boots (2018). "Convergence of Value Aggregation for Imitation Learning". In: *International Conference on Artificial Intelligence and Statistics*. Vol. 84, pp. 1801–1809.

Cheng, Ching-An and Han-Pang Huang (2016). "Learn the Lagrangian: A Vector-Valued RKHS Approach to Identifying Lagrangian Systems". In: *IEEE Transactions on Cybernetics* 46.12, pp. 3247–3258.

Cheng, Ching-An, Xinyan Yan, and Byron Boots (2019). "Trajectory-wise Control Variates for Variance Reduction in Policy Gradient Methods". In: *Conference on Robot Learning*.

Cheng, Ching-An, Han-Pang Huang, Huan-Kun Hsu, Wei-Zh Lai, and Chih-Chun Cheng (2016). "Learning the inverse dynamics of robotic manipulators in structured reproducing kernel Hilbert space". In: *IEEE Transactions on Cybernetics* 46.7, pp. 1691–1703.

Cheng, Ching-An, Xinyan Yan, Nolan Wagener, and Byron Boots (2018a). "Fast Policy Learning using Imitation and Reinforcement". In: *Conference on Uncertainty in Artificial Intelligence*.

Cheng, Ching-An, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff (2018b). "RMPflow: A Computational Graph for Automatic Motion Policy Generation". In: *The 13th International Workshop on the Algorithmic Foundations of Robotics*.

— (2018c). "Rmpflow: A computational graph for automatic motion policy generation". In: *arXiv preprint arXiv:1811.07049*.

Cheng, Ching-An, Remi Tachet des Combes, Byron Boots, and Geoff Gordon (2019a). "A Reduction from Reinforcement Learning to No-Regret Online Learning". In: *arXiv preprint arXiv:1911.05873*.

Cheng, Ching-An, Xinyan Yan, Evangelos A Theodorou, and Byron Boots (2019b). "Accelerating imitation learning with predictive models". In: *International Conference on Artificial Intelligence and Statistics*.

Cheng, Ching-An, Jonathan Lee, Ken Goldberg, and Byron Boots (2019c). "Online Learning with Continuous Variations: Dynamic Regret and Reductions". In: *arXiv preprint arXiv:1902.07286*.

Cheng, Ching-An, Xinyan Yan, Nathan Ratliff, and Byron Boots (2019d). "Predictor-Corrector Policy Optimization". In: *International Conference on Machine Learning*.

Cheng, Ching-An, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff (2019e). "RMPflow: A Geometric Framework for Generation of

Multi-Task Motion Policies". In: *IEEE Transactions on Automation Science and Engineering (in submission)*.

Chiang, Chao-Kai, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu (2012). "Online optimization with gradual variations". In: *Conference on Learning Theory*, pp. 6–1.

Correll, Nikolaus, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman (2018). "Analysis and observations from the first amazon picking challenge". In: *IEEE Transactions on Automation Science and Engineering* 15.1, pp. 172–188.

Cucker, Felipe and Ding Xuan Zhou (2007). *Learning theory: an approximation theory viewpoint*. Vol. 24. Cambridge University Press.

Daafouz, Jamal, Pierre Riedinger, and Claude Iung (2002). "Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach". In: *IEEE Transactions on Automatic Control* 47.11, pp. 1883–1887.

Dai, Bo, Albert Shaw, Niao He, Lihong Li, and Le Song (2018). "Boosting the actor with dual critic". In: *International Conference on Learning Representation*.

Dang, Cong D and Guanghui Lan (2015). "On the convergence properties of non-euclidean extragradient methods for variational inequalities with generalized monotone operators". In: *Computational Optimization and applications* 60.2, pp. 277–310.

Daniely, Amit, Alon Gonen, and Shai Shalev-Shwartz (2015). "Strongly adaptive online learning". In: *International Conference on Machine Learning*, pp. 1405–1411.

Daskalakis, Constantinos, Paul W Goldberg, and Christos H Papadimitriou (2009). "The complexity of computing a Nash equilibrium". In: *SIAM Journal on Computing* 39.1, pp. 195–259.

Deisenroth, Marc and Carl E Rasmussen (2011). "PILCO: A model-based and data-efficient approach to policy search". In: *International Conference on Machine Learning*, pp. 465–472.

Dellaert, Frank, Michael Kaess, et al. (2017). "Factor graphs for robot perception". In: *Foundations and Trends® in Robotics* 6.1-2, pp. 1–139.

Denardo, Eric V and Bennett L Fox (1968). "Multichain Markov renewal programs". In: *SIAM Journal on Applied Mathematics* 16.3, pp. 468–487.

Dhariwal, Prafulla, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu (2017). *OpenAI Baselines*.

Diakonikolas, Jelena and Lorenzo Orecchia (2017). "Accelerated extra-gradient descent: A novel accelerated first-order method". In: *arXiv preprint arXiv:1706.04680*.

Dietrich, Alexander, Alin Albu-Schäffer, and Gerd Hirzinger (2012). "On continuous null space projections for torque-based, hierarchical, multi-objective manipulation". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2978–2985.

Dietrich, Alexander, Christian Ott, and Jaeheung Park (2018). "The hierarchical operational space formulation: stability analysis for the regulation case". In: *IEEE Robotics and Automation Letters* 3.2, pp. 1120–1127.

Dixit, Rishabh, Amrit Singh Bedi, Ruchi Tripathi, and Ketan Rajawat (2019). "Online learning with inexact proximal online gradient descent algorithms". In: *IEEE Transactions on Signal Processing* 67.5, pp. 1338–1352.

Dong, J., M. Mukadam, F. Dellaert, and B. Boots (2016). "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs". In: *Robotics: Science and Systems*.

Drews, Paul, Grady Williams, Brian Goldfain, Evangelos A Theodorou, and James M Rehg (2017). "Aggressive Deep Driving: Model Predictive Control with a CNN Cost Model". In: *Conference on Robot Learning*, pp. 133–142.

Duan, Yan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel (2016). "Benchmarking deep reinforcement learning for continuous control". In: *International Conference on Machine Learning*, pp. 1329–1338.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul, pp. 2121–2159.

Erez, Tom, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svetoslav Kolev, and Emanuel Todorov (2013). "An integrated system for real-time model-predictive control of humanoid robots". In: *IEEE/RAS International Conference on Humanoid Robots*.

Escande, Adrien, Nicolas Mansard, and Pierre-Brice Wieber (2014). "Hierarchical quadratic programming: Fast online humanoid-robot motion generation". In: *The International Journal of Robotics Research* 33.7, pp. 1006–1028.

Facchinei, Francisco and Jong-Shi Pang (2007). *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media.

Featherstone, Roy (2008). *Rigid Body Dynamics Algorithms*. Springer.

Flacco, F., T. Kröger, A. De Luca, and O. Khatib (2012). "A depth space approach to human-robot collision avoidance". In: *IEEE International Conference on Robotics and Automation*, pp. 338–345.

Gammell, Jonathan D., Siddhartha S. Srinivasa, and Timothy D. Barfoot (2015). "Batch Informed Trees (BIT*): Sampling-based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs". In: *IEEE International Conference on Robotics and Automation*.

Garber, Dan and Elad Hazan (2015). "Faster rates for the frank-wolfe method over strongly-convex sets". In: *International Conference on Machine Learning*.

Garcıa, Javier and Fernando Fernández (2015). "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1, pp. 1437–1480.

Geist, A René, Andreas Hansen, Eugen Solowjow, Shun Yang, and Edwin Kreuzer (2017). "Data Collection for Robust End-to-End Lateral Vehicle Control". In: *ASME 2017 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, V001T45A007–V001T45A007.

Ghadimi, Saeed, Guanghui Lan, and Hongchao Zhang (2016). "Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization". In: *Mathematical Programming* 155.1-2, pp. 267–305.

Gibbs, Alison L and Francis Edward Su (2002). "On choosing and bounding probability metrics". In: *International Statistical Review* 70.3, pp. 419–435.

Gijsberts, Arjan and Giorgio Metta (2013). "Real-time model learning using incremental sparse spectrum gaussian process regression". In: *Neural Networks* 41, pp. 59–69.

Gordon, Geoffrey J (1995). "Stable function approximation in dynamic programming". In: *Machine Learning Proceedings 1995*. Elsevier, pp. 261–268.

— (1999). "Regret bounds for prediction problems". In: *Conference on Learning Theory*. Vol. 99, pp. 29–40.

Grathwohl, Will, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud (2018). "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation". In: *International Conference on Learning Representations*.

Greensmith, Evan, Peter L Bartlett, and Jonathan Baxter (2004). "Variance reduction techniques for gradient estimates in reinforcement learning". In: *Journal of Machine Learning Research* 5.Nov, pp. 1471–1530.

Gupta, Vineet, Tomer Koren, and Yoram Singer (2017). "A unified approach to adaptive regularization in online and stochastic optimization". In: *arXiv preprint arXiv:1706.06569*.

Hall, Eric and Rebecca Willett (2013). "Dynamical Models and tracking regret in online convex programming". In: *International Conference on Machine Learning*, pp. 579–587.

Hayes, Thomas P (2005). "A large-deviation inequality for vector-valued martingales". In: *Combinatorics, Probability and Computing*.

Hazan, Elad, Amit Agarwal, and Satyen Kale (2007). "Logarithmic regret algorithms for online convex optimization". In: *Machine Learning* 69.2, pp. 169–192.

Hazan, Elad et al. (2016). "Introduction to online convex optimization". In: *Foundations and Trends® in Optimization* 2.3-4, pp. 157–325.

Hensman, James, Nicolo Fusi, and Neil D Lawrence (2013). "Gaussian processes for big data". In: *arXiv preprint arXiv:1309.6835*.

Hernández-Lerma, Onésimo and Jean B Lasserre (2012). *Discrete-time Markov control processes: basic optimality criteria*. Vol. 30. Springer Science & Business Media.

Ho-Nguyen, Nam and Fatma Kılınç-Karzan (2018). "Exploiting problem structure in optimization under uncertainty via online convex optimization". In: *Mathematical Programming*, pp. 1–35.

Ijspeert, A. J., J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal (2013). "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors". In: *Neural Computation* 25.2, pp. 328–373.

Ivan, Vladimir, Dmitry Zarubin, Marc Toussaint, Taku Komura, and Sethu Vijayakumar (2013). "Topology-based representations for motion planning and generalization in dynamic environments with interactions". In: *The International Journal of Robotics Research* 32.9-10, pp. 1151–1163.

Jacobson, David H and David Q Mayne (1970). "Differential dynamic programming". In:

Jadbabaie, Ali, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan (2015). "Online optimization: Competing with dynamic comparators". In: *Artificial Intelligence and Statistics*, pp. 398–406.

Jaggi, Martin (2013). "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization." In: *International Conference on Machine Learning*, pp. 427–435.

Jofré, Alejandro and Roger J-B Wets (2014). "Variational convergence of bifunctions: motivating applications". In: *SIAM Journal on Optimization* 24.4, pp. 1952–1979.

Johnson, Matthew, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, et al. (2015). "Team IHMC's lessons learned from the DARPA robotics challenge trials". In: *Journal of Field Robotics* 32.2, pp. 192–208.

Journée, Michel, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre (2010). "Generalized power method for sparse principal component analysis". In: *Journal of Machine Learning Research* 11.Feb, pp. 517–553.

Juditsky, Anatoli, Arkadi Nemirovski, and Claire Tauvel (2011). "Solving variational inequalities with stochastic mirror-prox algorithm". In: *Stochastic Systems* 1.1, pp. 17–58.

Juditsky, Anatoli, Arkadi Nemirovski, et al. (2011). "First order methods for nonsmooth convex large-scale optimization, i: general purpose methods". In: *Optimization for Machine Learning*, pp. 121–148.

Kaess, Michael, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert (2012). "iSAM2: Incremental smoothing and mapping using the Bayes tree". In: *The International Journal of Robotics Research* 31.2, pp. 216–235.

Kahn, Gregory, Tianhao Zhang, Sergey Levine, and Pieter Abbeel (2017). "Plato: Policy learning using adaptive trajectory optimization". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 3342–3349.

Kakade, Sham and John Langford (2002). "Approximately optimal approximate reinforcement learning". In: *International Conference on Machine Learning*. Vol. 2, pp. 267–274.

Kakade, Sham M (2002). "A natural policy gradient". In: *Advances in Neural Information Processing Systems*, pp. 1531–1538.

Kakade, Sham M and Ambuj Tewari (2009). "On the generalization ability of online strongly convex programming algorithms". In: *Advances in Neural Information Processing Systems*, pp. 801–808.

Kakade, Sham Machandranath et al. (2003). "On the sample complexity of reinforcement learning". PhD thesis. University of London London, England.

Kalai, Adam and Santosh Vempala (2005). "Efficient algorithms for online decision problems". In: *Journal of Computer and System Sciences* 71.3, pp. 291–307.

Kaldestad, K. B., S. Haddadin, R. Belder, G. Hovland, and D. A. Anisi (2014). "Collision avoidance with potential fields based on parallel processing of 3D-point cloud data on the GPU". In: *IEEE International Conference on Robotics and Automation*, pp. 3250–3257.

Kappler, Daniel, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg (2018). "Real-time Perception meets Reactive Motion Generation". In: *IEEE Robotics and Automation Letters* 3.3, pp. 1864–1871.

Karaman, Sertac and Emilio Frazzoli (2011). "Sampling-based Algorithms for Optimal Motion Planning". In: *The International Journal of Robotics Research* 30.7, pp. 846–894.

Kearns, Michael J and Satinder P Singh (1999). "Finite-sample convergence rates for Q-learning and indirect algorithms". In: *Advances in Neural Information Processing Systems*, pp. 996–1002.

Khalil, Hassan K (1996). "Noninear systems". In: *Prentice-Hall, New Jersey* 2.5, pp. 5–1.

Khatib, O. (1985). "Real-time obstacle avoidance for manipulators and mobile robots". In: *IEEE International Conference on Robotics and Automation*. Vol. 2, pp. 500–505.

Khatib, Oussama (1987). "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal on Robotics and Automation* 3.1, pp. 43–53.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Koenig, Nathan and Andrew Howard (2004). "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *IEEE/RSJ International Conference onIntelligent Robots and Systems*. Vol. 3. IEEE, pp. 2149–2154.

Komlósi, SÁNDOR (1999). "On the Stampacchia and Minty variational inequalities". In: *Generalized Convexity and Optimization for Economic and Financial Decisions*, pp. 231–260.

Konda, Vijay R and John N Tsitsiklis (2000). "Actor-critic algorithms". In: *Advances in Neural Information Processing Systems*, pp. 1008–1014.

Kong, Jason, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli (2015). "Kinematic and dynamic vehicle models for autonomous driving control design". In: *IEEE Intelligent Vehicles Symposium*. IEEE, pp. 1094–1099.

Konnov, Igor V (2007). “Combined relaxation methods for generalized monotone variational inequalities”. In: *Generalized convexity and related topics*. Springer, pp. 3–31.

Konnov, Igor V and Erkki Laitinen (2002). *Theory and applications of variational inequalities*. University of Oulu, Department of Mathematical Sciences.

Konnov, IV and S Schaible (2000). “Duality for equilibrium problems under generalized monotonicity”. In: *Journal of Optimization Theory and Applications* 104.2, pp. 395–408.

Kontorovich, Aryeh and Maxim Raginsky (2017). “Concentration of measure without independence: a unified approach via the martingale method”. In: *Convexity and Concentration*. Springer, pp. 183–210.

Korpelevich, GM (1976). “The extragradient method for finding saddle points and other problems”. In: *Matecon* 12, pp. 747–756.

Lakshminarayanan, Chandrashekar, Shalabh Bhatnagar, and Csaba Szepesvári (2018). “A linearly relaxed approximate linear program for Markov decision processes”. In: *IEEE Transactions on Automatic Control* 63.4, pp. 1185–1191.

Lan, Guanghui (2013). “The complexity of large-scale convex programming under a linear optimization oracle”. In: *arXiv preprint arXiv:1309.5550*.

Laskey, Michael, Caleb Chuck, Jonathan Lee, Jeffrey Mahler, Sanjay Krishnan, Kevin Jamieson, Anca Dragan, and Ken Goldberg (2016). “Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations”. In: *arXiv preprint arXiv:1610.00850*.

— (2017). “Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations”. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 358–365.

Laurense, Vincent A, Jonathan Y Goh, and J Christian Gerdes (2017). “Path-tracking for autonomous vehicles at the limit of friction”. In: *2017 American Control Conference (ACC)*, pp. 5586–5591.

LaValle, Steven M. (2006). *Planning Algorithms*. Available at http://planning.cs.uiuc.edu/. Cambridge, U.K.: Cambridge University Press.

Lázaro-Gredilla, Miguel, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R. Figueiras-Vidal (2010). “Sparse spectrum Gaussian process regression”. In: *Journal of Machine Learning Research* 11.Jun, pp. 1865–1881.

Le, Hoang M, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III (2018). "Hierarchical Imitation and Reinforcement Learning". In: *arXiv preprint arXiv:1803.00590*.

Lee, Donghwan and Niao He (2018). "Stochastic Primal-Dual Q-Learning". In: *arXiv preprint arXiv:1810.08298*.

Lee, Jaemin, Nicolas Mansard, and Jaeheung Park (2012). "Intermediate desired value approach for task transition of robots in kinematic control". In: *IEEE Transactions on Robotics* 28.6, pp. 1260–1277.

Lee, Jeffrey M. (2009). *Manifolds and differential geometry*. Graduate Studies in Mathematics, vol. 107, American Mathematical Society.

Lee, Jeongseok, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu (2018a). "DART: Dynamic Animation and Robotics Toolkit". In: *The Journal of Open Source Software* 3.22, p. 500.

Lee, Jeongseok, Michael X Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S Srinivasa, Mike Stilman, and C Karen Liu (2018b). "DART: Dynamic animation and robotics toolkit". In: *The Journal of Open Source Software* 3.22, p. 500.

Lee, Jonathan, Michael Laskey, Ajay Kumar Tanwani, Anil Aswani, and Ken Goldberg (2018c). "A Dynamic Regret Analysis and Adaptive Regularization Algorithm for On-Policy Robot Imitation Learning". In: *arXiv preprint arXiv:1811.02184*.

Lee, Wee Sun, Peter L Bartlett, and Robert C Williamson (1998). "The importance of convexity in learning with squared loss". In: *IEEE Transactions on Information Theory* 44.5, pp. 1974–1980.

Levine, Sergey and Vladlen Koltun (2013). "Guided policy search". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1–9.

Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel (Jan. 2016). "End-to-end Training of Deep Visuomotor Policies". In: *Journal of Machine Learning Research* 17.1, pp. 1334–1373.

Lewis, Andrew D (2000). *The geometry of the maximum principle for affine connection control systems*.

Li, Anqi, Mustafa Mukadam, Magnus Egerstedt, and Byron Boots (2019a). "Multi-Objective Policy Generation for Multi-Robot Systems Using Riemannian Motion Policies". In: *International Symposium on Robotics Research*.

Li, Anqi, Ching-An Cheng, Byron Boots, and Magnus Egerstedt (2019b). "Stable, Concurrent Controller Composition for Multi-Objective Robotic Tasks". In: *IEEE Conference on Decision and Control*.

Liberzon, Daniel, Joao P Hespanha, and A Stephen Morse (1999). "Stability of switched systems: a Lie-algebraic condition". In: *Systems & Control Letters* 37.3, pp. 117–122.

Liegeois, Alain (1977). "Automatic supervisory control of the configuration and behaviour of multibody mechanisms". In: *IEEE Transactions on Systems, Man and Cybernetics* 7.12, pp. 868–871.

Lin, Qihang, Selvaprabu Nadarajah, and Negar Soheili (2017). *Revisiting approximate linear programming using a saddle point based reformulation and root finding solution approach*. Tech. rep. working paper, U. of Il. at Chicago and U. of Iowa.

Lin, Qihang, Mingrui Liu, Hassan Rafique, and Tianbao Yang (2018). "Solving Weakly-Convex-Weakly-Concave Saddle-Point Problems as Weakly-Monotone Variational Inequality". In: *arXiv preprint arXiv:1810.10207*.

Ljung, Lennart (1998). "System identification". In: *Signal analysis and prediction*. Springer, pp. 163–173.

Lo, Sheng-Yen, Ching-An Cheng, and Han-Pang Huang (2016). "Virtual impedance control for safe human-robot interaction". In: *Journal of Intelligent & Robotic Systems* 82.1, pp. 3–19.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Mainprice, Jim, Nathan Ratliff, and Stefan Schaal (2016). "Warping the Workspace Geometry with Electric Potentials for Motion Optimization of Manipulation Tasks". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Mann, W Robert (1953). "Mean value methods in iteration". In: *Proceedings of the American Mathematical Society* 4.3, pp. 506–510.

Manne, Alan S et al. (1959). *Linear programming and sequential decision models*. Tech. rep. Cowles Foundation for Research in Economics, Yale University.

McDiarmid, Colin (1998). "Concentration". In: *Probabilistic methods for algorithmic discrete mathematics*. Springer, pp. 195–248.

McMahan, H Brendan (2017). "A survey of algorithms and analysis for adaptive online learning". In: *The Journal of Machine Learning Research* 18.1, pp. 3117–3166.

McMahan, H Brendan and Matthew Streeter (2010). "Adaptive bound optimization for online convex optimization". In: *arXiv preprint arXiv:1002.4908*.

Meng, Xiangyun, Nathan Ratliff, Yu Xiang, and Dieter Fox (2019). "Learning Latent Space Dynamics for Tactile Servoing". In: *IEEE International Conference on Robotics and Automation*.

Michels, Jeff, Ashutosh Saxena, and Andrew Y Ng (2005). "High speed obstacle avoidance using monocular vision and reinforcement learning". In: *International Conference on Machine learning*, pp. 593–600.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602*.

Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016). "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning*, pp. 1928–1937.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2012). *Foundations of machine learning*. MIT press.

Mokhtari, Aryan, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro (2016). "Online optimization in dynamic environments: Improved regret rates for strongly convex problems". In: *Conference on Decision and Control*. IEEE, pp. 7195–7201.

Mordatch, Igor and Emo Todorov (2014). "Combining the benefits of function approximation and trajectory optimization." In: *Robotics: Science and Systems*, pp. 5–32.

Morris, Benjamin, Matthew J Powell, and Aaron D Ames (2013). "Sufficient conditions for the lipschitz continuity of qp-based multi-objective control of humanoid robots". In: *IEEE Conference on Decision and Control*. IEEE, pp. 2920–2926.

Mukadam, Mustafa, Xinyan Yan, and Byron Boots (2016). "Gaussian Process Motion Planning." In: *IEEE International Conference on Robotics and Automation*.

Mukadam, Mustafa, Ching-An Cheng, Xinyan Yan, and Byron Boots (2017). "Approximately Optimal Continuous-Time Motion Planning and Control via Probabilistic Inference". In: *IEEE International Conference on Robotics and Automation*, pp. 664–671.

Mukadam, Mustafa, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots (2018). "Continuous-time Gaussian process motion planning via probabilistic inference". In: *The International Journal of Robotics Research* 37.11, pp. 1319–1340.

Mukadam, Mustafa, Ching-An Cheng, Dieter Fox, Byron Boots, and Nathan Ratliff (2019). "Riemannian Motion Policy Fusion through Learnable Hierarchical Energy Reshaping". In: *Conference on Robot Learning*.

Muller, Urs, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L Cun (2006). "Off-road obstacle avoidance through end-to-end learning". In: *Advances in Neural Information Processing Systems*, pp. 739–746.

Nair, Ashvin, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel (2017). "Overcoming exploration in reinforcement learning with demonstrations". In: *arXiv preprint arXiv:1709.10089*.

Nakanishi, J., R. Cory, M. Mistry, J. Peters, and S. Schaal (2008). "Operational space control: A theoretical and empirical comparison". In: *The International Journal of Robotics Research* 6, pp. 737–757.

Narendra, Kumpati S and Jeyendran Balakrishnan (1994). "A common Lyapunov function for stable LTI systems with commuting A-matrices". In: *IEEE Transactions on Automatic Control* 39.12, pp. 2469–2471.

Nash, John (1956). "The imbedding problem for Riemannian manifolds". In: *Annals of Mathematics* 63.1, pp. 20–63.

Nemirovski, Arkadi (2004). "Prox-method with rate of convergence O (1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems". In: *SIAM Journal on Optimization* 15.1, pp. 229–251.

Nemirovski, Arkadi, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro (2009). "Robust stochastic approximation approach to stochastic programming". In: *SIAM Journal on Optimization* 19.4, pp. 1574–1609.

Nesterov, Yurii (2013). *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media.

Ng, Andrew Y, Daishi Harada, and Stuart Russell (1999). "Policy invariance under reward transformations: Theory and application to reward shaping". In: *International Conference on Machine Learning*. Vol. 99, pp. 278–287.

Oh, Junhyuk, Satinder Singh, and Honglak Lee (2017). "Value prediction network". In: *Advances in Neural Information Processing Systems*, pp. 6120–6130.

Paden, Brian, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli (2016). "A survey of motion planning and control techniques for self-driving urban vehicles". In: *IEEE Transactions on Intelligent Vehicles* 1.1, pp. 33–55.

Pan, Y., Ching-An Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots. (2019). "Imitation Learning for Agile Autonomous Driving". In: *The International Journal of Robotics Research*.

Pan, Yunpeng and Evangelos Theodorou (2014). "Probabilistic differential dynamic programming". In: *Advances in Neural Information Processing Systems*, pp. 1907–1915.

Pan, Yunpeng, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots (2017a). "Agile Off-Road Autonomous Driving Using End-to-End Deep Imitation Learning". In: *arXiv preprint arXiv:1709.07174*.

Pan, Yunpeng, Xinyan Yan, Evangelos A. Theodorou, and Byron Boots (2017b). "Prediction under Uncertainty in Sparse Spectrum Gaussian Processes with Applications to Filtering and Control". In: *International Conference on Machine Learning*, pp. 2760–2768.

Pan, Yunpeng, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots (2018). "Agile Off-Road Autonomous Driving Using End-to-End Deep Imitation Learning". In: *Robotics: Science and Systems*.

Papini, Matteo, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli (2018). "Stochastic Variance-Reduced Policy Gradient". In: *International Conference on Machine Learning*, pp. 4023–4032.

Pascanu, Razvan, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sebastien Racanière, David Reichert, Théophane Weber, Daan Wierstra, and Peter Battaglia (2017). "Learning model-based planning from scratch". In: *arXiv preprint arXiv:1707.06170*.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). "Automatic differentiation in pytorch". In:

Paxton, Chris, Nathan Ratliff, Clemens Eppner, and Dieter Fox (2019). "Representing Robot Task Plans as Robust Logical-Dynamical Systems". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Peng, Xue Bin, Pieter Abbeel, Sergey Levine, and Michiel van de Panne (2018). "DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills". In: *arXiv preprint arXiv:1804.02717*.

Peters, Jan, Katharina Mülling, and Yasemin Altun (2010). "Relative Entropy Policy Search." In: *Conference on Artificial Intelligence*. Atlanta, pp. 1607–1612.

Peters, Jan and Stefan Schaal (2008). "Natural actor-critic". In: *Neurocomputing* 71.7-9, pp. 1180–1190.

Peters, Jan, Michael Mistry, Firdaus Udwadia, Jun Nakanishi, and Stefan Schaal (2008). "A unifying framework for robot control with redundant DOFs". In: *Autonomous Robots* 24.1, pp. 1–12.

Pickem, Daniel, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt (2017). "The Robotarium: A remotely accessible swarm robotics research testbed". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 1699–1706.

Platt, Robert, Muhammad E Abdallah, and Charles W Wampler (2011). "Multiple-priority impedance control." In: *IEEE International Conference on Robotics and Automation*, pp. 6033–6038.

Pomerleau, Dean A (1989). "Alvinn: An autonomous land vehicle in a neural network". In: *Advances in Neural Information Processing Systems*, pp. 305–313.

Puterman, Martin L (2014). *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Rajamani, Rajesh (2011). *Vehicle dynamics and control*. Springer Science & Business Media.

Rajeswaran, Aravind, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine (2017). "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations". In: *arXiv preprint arXiv:1709.10087*.

Rakhlin, Alexander and Karthik Sridharan (2012). "Online learning with predictable sequences". In: *arXiv preprint arXiv:1208.3728*.

Rakhlin, Sasha and Karthik Sridharan (2013). "Optimization, learning, and games with predictable sequences". In: *Advances in Neural Information Processing Systems*, pp. 3066–3074.

Ratliff, Nathan, Marc Toussaint, and Stefan Schaal (2015a). "Understanding the geometry of workspace obstacles in motion optimization". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 4202–4209.

— (2015b). "Understanding the Geometry of Workspace Obstacles in Motion Optimization". In: *IEEE International Conference on Robotics and Automation*.

Ratliff, Nathan, Matthew Zucker, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa (2009). "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning". In: *IEEE International Conference on Robotics and Automation*.

Ratliff, Nathan D, Jan Issac, and Daniel Kappler (2018). “Riemannian Motion Policies”. In: *arXiv preprint arXiv:1801.02854*.

Rausch, Viktor, Andreas Hansen, Eugen Solowjow, Chang Liu, Edwin Kreuzer, and J. Karl Hedrick (2017). “Learning a Deep Neural Net Policy for End-to-End Control of Autonomous Vehicles”. In: *IEEE American Control Conference*.

Rawlik, Konrad, Marc Toussaint, and Sethu Vijayakumar (2012). “On stochastic optimal control and reinforcement learning by approximate inference”. In: *Robotics: Science and Systems*. Vol. 13. 2, pp. 3052–3056.

Reddi, Sashank J, Satyen Kale, and Sanjiv Kumar (2018). “On the convergence of adam and beyond”. In: *International Conference on Learning Representations*.

Riedmiller, Martin (2005). “Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method”. In: *European Conference on Machine Learning*. Springer, pp. 317–328.

Rimon, E. and D.E. Koditschek (1991). “The Construction of Analytic Diffeomorphisms for Exact Robot Navigation on Star Worlds”. In: *Transactions of the American Mathematical Society* 327.1, pp. 71–116.

Ross, Stephane and J Andrew Bagnell (2012). “Agnostic system identification for model-based reinforcement learning”. In:

— (2014). “Reinforcement and imitation learning via interactive no-regret learning”. In: *arXiv preprint arXiv:1406.5979*.

Ross, Stéphane, Geoffrey J Gordon, and Drew Bagnell (2011). “A reduction of imitation learning and structured prediction to no-regret online learning”. In: *International Conference on Artificial Intelligence and Statistics*, pp. 627–635.

Ross, Stéphane, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert (2013). “Learning monocular reactive uav control in cluttered natural environments”. In: *IEEE International Conference onRobotics and Automation*. IEEE, pp. 1765–1772.

Rutherford, Simon J and David J Cole (2010). “Modelling nonlinear vehicle dynamics with neural networks”. In: *International journal of vehicle design* 53.4, pp. 260–287.

Salimbeni, Hugh, Ching-An Cheng, Byron Boots, and Marc Deisenroth (2018). “Orthogonally Decoupled Variational Gaussian Processes”. In: *Conference on Neural Information Processing Systems*.

Schaal, Stefan (1999). “Is imitation learning the route to humanoid robots?” In: *Trends in Cognitive Sciences* 3.6, pp. 233–242.

Schmidt, T., R. Newcombe, and D. Fox (2015). “DART: Dense articulated real-time tracking with consumer depth cameras”. In: *Autonomous Robots* 39.3.

Schroecker, Yannick and Charles L Isbell (2017). “State Aware Imitation Learning”. In: *Advances in Neural Information Processing Systems*, pp. 2911–2920.

Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel (2015a). “High-dimensional continuous control using generalized advantage estimation”. In: *arXiv preprint arXiv:1506.02438*.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz (2015b). “Trust region policy optimization”. In: *International Conference on Machine Learning*, pp. 1889–1897.

Sentis, Luis and Oussama Khatib (2006). “A whole-body control framework for humanoids operating in human environments”. In: *IEEE International Conference on Robotics and Automation*, pp. 2641–2648.

Shaban, Amirreza, Ching-An Cheng, Nathan Hatch, and Byron Boots (2019). “Truncated Back Propagation for Bilevel Optimization”. In: *International Conference on Artificial Intelligence and Statistics*.

Shalev-Shwartz, Shai (2012). “Online learning and online convex optimization”. In: *Foundations and Trends® in Machine Learning* 4.2, pp. 107–194.

Shalev-Shwartz, Shai, Ohad Shamir, Nathan Srebro, and Karthik Sridharan (2009). “Stochastic Convex Optimization.” In: *Conference on Learning Theory*.

Shimodaira, Hidetoshi (2000). “Improving predictive inference under covariate shift by weighting the log-likelihood function”. In: *Journal of statistical planning and inference* 90.2, pp. 227–244.

Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller (2014). “Deterministic policy gradient algorithms”. In: *International Conference on Machine Learning*.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. (2017a). “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676, pp. 354–359.

Silver, David, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. (2017b). "The predictron: End-to-end learning and planning". In: *International Conference on Machine Learning*.

Slotine, Siciliano B (1991). "A general framework for managing multiple tasks in highly redundant robotic systems". In: *Proceeding of 5th International Conference on Advanced Robotics*. Vol. 2, pp. 1211–1216.

Snelson, Edward and Zoubin Ghahramani (2006). "Sparse Gaussian processes using pseudo-inputs". In: *Advances in Neural Information Processing Systems*.

Sontag, Eduardo D (1983). "A Lyapunov-like characterization of asymptotic controllability". In: *SIAM Journal on Control and Optimization* 21.3, pp. 462–471.

Squires, Eric, Pietro Pierpaoli, and Magnus Egerstedt (2018). "Constructive Barrier Certificates with Applications to Fixed-Wing Aircraft Collision Avoidance". In: *Conference on Control Technology and Applications*. IEEE, pp. 1656–1661.

Srinivas, Aravind, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn (2018). "Universal Planning Networks: Learning Generalizable Representations for Visuomotor Control". In: *Proceedings of the 35th International Conference on Machine Learning*.

Sun, Wen, J. Andrew Bagnell, and Byron Boots (2018). "Truncated Horizon Policy Search: Deep Combination of Reinforcement and Imitation". In: *International Conference on Learning Representations*.

Sun, Wen, Arun Venkatraman, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell (2017). "Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction". In: *International Conference on Machine Learning*, pp. 3309–3318.

Sun, Wen, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell (2018). "Dual Policy Iteration". In: *Advances in Neural Information Processing Systems 31*, pp. 7059–7069.

Sutanto, Giovanni, Nathan Ratliff, Balakumar Sundaralingam, Yevgen Chebotar, Zhe Su, Ankur Handa, and Dieter Fox (2019). "Learning Latent Space Dynamics for Tactile Servoing". In: *IEEE International Conference on Robotics and Automation*.

Sutton, Richard S (1991). "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM SIGART Bulletin* 2.4, pp. 160–163.

Sutton, Richard S and Andrew G Barto (1998). *Introduction to reinforcement learning*. Vol. 135. MIT Press Cambridge.

Sutton, Richard S, David A McAllester, Satinder P Singh, and Yishay Mansour (2000). "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in Neural Information Processing Systems*, pp. 1057–1063.

Sutton, Richard S, Csaba Szepesvári, Alborz Geramifard, and Michael P Bowling (2012). "Dyna-style planning with linear function approximation and prioritized sweeping". In: *arXiv preprint arXiv:1206.3285*.

Tan, Jie, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke (2018). "Sim-to-real: Learning agile locomotion for quadruped robots". In: *arXiv preprint arXiv:1804.10332*.

Tassa, Yuval, Tom Erez, and William D Smart (2008). "Receding horizon differential dynamic programming". In: *Advances in Neural Information Processing Systems*, pp. 1465–1472.

Taylor, John R. (2005). *Classical Mechanics*. University Science Books.

Tieleman, Tijmen and Geoffrey Hinton (2012). "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2, pp. 26–31.

Titsias, Michalis K. (2009). "Variational learning of inducing variables in sparse Gaussian processes". In: *International Conference on Artificial Intelligence and Statistics*, pp. 567–574.

Todorov, E. (2006). "Optimal control theory". In: *In Bayesian Brain: Probabilistic Approaches to Neural Coding*. Ed. by K. et al. Doya, pp. 269–298.

Todorov, Emanuel and Weiwei Li (2005). "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear systems". In: *American Control Conference*. IEEE, pp. 300–306.

Toussaint, Marc (2009). "Robot Trajectory Optimization using Approximate Inference". In: *International Conference on Machine Learning*, pp. 1049–1056. ISBN: 978-1-60558-516-1.

Tucker, George, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein (2017). "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models". In: *Advances in Neural Information Processing Systems*, pp. 2624–2633.

Udwadia, F. E. (2003). "A new perspective on the tracking control of nonlinear structural and mechanical systems". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 459.2035, pp. 1783–1800. eprint: `http:`

//rspa.royalsocietypublishing.org/content/459/2035/1783.full.pdf.

Udwadia, F. E. and R. E. Kalaba (1996). *Analytical Dynamics: A New Approach*. Cambridge University Press.

Urmson, Chris, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. (2008). "Autonomous driving in urban environments: Boss and the urban challenge". In: *Journal of Field Robotics* 25.8, pp. 425–466.

Vapnik, Vladimir Naumovich (1998). *Statistical learning theory*. Vol. 1. Wiley New York.

Veliov, VM and Phan Tu Vuong (2017). "Gradient Methods on Strongly Convex Feasible Sets and Optimal Control of Affine Systems". In: *Applied Mathematics & Optimization*, pp. 1–34.

Venkatraman, Arun, Martial Hebert, and J Andrew Bagnell (2015). "Improving multi-step prediction of learned time series models". In: *Conference on Artificial Intelligence*.

Volodymyr, Mnih, Kavukcuoglu Koray, Silver David, A Rusu Andrei, and Veness Joel (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533.

Vu, Linh and Daniel Liberzon (2005). "Common Lyapunov functions for families of commuting nonlinear systems". In: *Systems & Control Letters* 54.5, pp. 405–416.

Wagener, Nolan, Ching-An Cheng, Jacob Sacks, and Byron Boots (2019). "An Online Learning Approach to Model Predictive Control". In: *Robotics: Science and Systems*.

Walker, Michael W and David E Orin (1982). "Efficient dynamic computer simulation of robotic mechanisms". In: *Journal of Dynamic Systems, Measurement, and Control* 104.3, pp. 205–211.

Wang, Li, Aaron D Ames, and Magnus Egerstedt (2016). "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots". In: *IEEE Conference on Decision and Control*. IEEE, pp. 2659–2664.

Wang, Mengdi (2017a). "Primal-Dual π Learning: Sample Complexity and Sublinear Run Time for Ergodic Markov Decision Problems". In: *arXiv preprint arXiv:1710.06100*.

— (2017b). "Randomized linear programming solves the discounted markov decision problem in nearly-linear (sometimes sublinear) running time". In: *arXiv preprint arXiv:1704.01869*.

Wang, Mengdi and Yichen Chen (2016). "An online primal-dual method for discounted Markov decision processes". In: *Conference on Decision and Control*. IEEE, pp. 4516–4521.

Watterson, Michael, Sikang Liu, Ke Sun, Trey Smith, and Vijay Kumar (2018). "Trajectory Optimization On Manifolds with Applications to $SO(3)$ and $\mathbb{R}^3 \times S^2$". In: *Robotics: Science and Systems*.

Williams, C.K.I and C.E. Rasmussen (2006). *Gaussian processes for machine learning*. MIT Press.

Williams, Grady, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou (2016). "Aggressive driving with model predictive path integral control". In: *IEEE International Conference on Robotics and Automation*, pp. 1433–1440.

Williams, Grady, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou (2017). "Information theoretic mpc for model-based reinforcement learning". In: *IEEE International Conference on Robotics and Automation*, pp. 1714–1721.

Williams, Ronald J (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Reinforcement Learning*. Springer, pp. 5–32.

Yang, Tianbao, Lijun Zhang, Rong Jin, and Jinfeng Yi (2016). "Tracking slowly moving clairvoyant: optimal dynamic regret of online learning with true and noisy gradient". In: *International Conference on Machine Learning*. JMLR. org, pp. 449–457.

Yang, Zhengyuan, Yixuan Zhang, Jerry Yu, Junjie Cai, and Jiebo Luo (2018). "End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perception". In: *arXiv preprint arXiv:1801.06734*.

Yu, Hao, Shu Yang, Weihao Gu, and Shaoyu Zhang (2017). "Baidu driving dataset and end-to-end reactive control model". In: *IEEE Intelligent Vehicles Symposium*, pp. 341–346.

Zeiler, Matthew D (2012). "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701*.

Zhang, Jiakai and Kyunghyun Cho (2016). "Query-efficient imitation learning for end-to-end autonomous driving". In: *arXiv preprint arXiv:1605.06450*.

Zhang, Lijun, Tianbao Yang, Jinfeng Yi, Jing Rong, and Zhi-Hua Zhou (2017). "Improved Dynamic Regret for Non-degenerate Functions". In: *Advances in Neural Information Processing Systems*, pp. 732–741.

Zinkevich, Martin (2003). “Online convex programming and generalized infinitesimal gradient ascent”. In: *International Conference on Machine Learning*, pp. 928–936.