Policy Gradient Reinforcement Learning Without Regret

by

Travis Dick

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

This thesis consists of two independent projects, each contributing to a central goal of artificial intelligence research: to build computer systems that are capable of performing tasks and solving problems without problem-specific direction from us, their designers. I focus on two formal learning problems that have a strong mathematical grounding. Many real-world learning problems can be cast as instances of one of these two problems. Whenever our translation from the real to the formal accurately captures the character of the problem, then the mathematical arguments we make about algorithms in the formal setting will approximately hold in the real-world as well.

The first project focuses on an open question in the theory of policy gradient reinforcement learning methods. These methods learn by trial and error and decide whether a trial was good or bad by comparing its outcome to a given baseline. The baseline has no impact on the formal asymptotic guarantees for policy gradient methods, but it does alter their finite-time behaviour. This immediately raises the question: which baseline should we use? I propose that the baseline should be chosen such that a certain estimate used internally by policy gradient methods has the smallest error. I prove that, under slightly idealistic assumptions, this baseline gives a good upper bound on the regret of policy gradient methods. I derive closed-form expressions for this baseline in terms of properties of the formal learning problem and the computer's behaviour. The quantities appearing in the

closed form expressions are often unknown, so I also propose two algorithms for estimating this baseline from only known quantities. Finally, I present an empirical comparison of commonly used baselines that demonstrates improved performance when using my proposed baseline.

The second project focuses on a recently proposed class of formal learning problems that is in the intersection of two fields of computing science research: reinforcement learning and online learning. The considered problems are sometimes called online Markov decision processes, or Markov decision processes with changing rewards. The unique property of this class is that it assumes the computer's environment is adversarial, as though it were playing a game against the computer. This is in contrast to the more common assumption that the environment's behaviour is determined entirely by stochastic models. I propose three new algorithms for learning in Markov decision processes with changing rewards under various conditions. I prove theoretical performance guarantees for each algorithm that either complement or improve the best existing results and that often hold even under weaker assumptions. This comes at the cost of increased (but still polynomial) computational complexity. Finally, in the development and analysis of these algorithms, it was necessary to analyze an approximate version of a well-known optimization algorithm called online mirror ascent. To the best of my knowledge, this is the first rigorous analysis of this algorithm and it is of independent interest.

Contents

Abstract ii

1 Introduction **1**

2 Reinforcement Learning and Decision Processes **5**
2.1 Markov decision processes 6
2.1.1 Total Reward in Episodic MDPs 9
2.1.2 Average Reward in Ergodic MDPs 11
2.2 Markov deicison processes with changing rewards 13
2.2.1 Loop-free Episodic MDPCRs 17
2.2.2 Uniformly Ergodic MDPCRs 19

3 Gradient Methods **21**
3.1 Gradient Ascent 22
3.2 Stochastic Gradient Ascent 25
3.3 Online Mirror Ascent 27

4 Policy Gradient Methods and Baseline Functions **32**
4.1 Policy Gradient Methods 33
4.2 Baseline Functions 36
4.3 MSE Minimizing Baseline 41
4.3.1 Regret Bound from the MSE Minimizing Baseline . . 42
4.3.2 MSE Minimizing Baseline for Average Reward 43
4.3.3 MSE Minimizing Baseline for Total Reward 45
4.4 Estimating the MSE Minimizing Baseline 50

4.5 Experiments 55

5 Learning in MDPCRs **69**

5.1 Reductions to Online Linear Optimization 70

5.1.1 Reduction of Loop-Free Episodic MDPCRs 71

5.1.2 Reduction of Uniformly Ergodic MDPCRs 75

5.2 Online Mirror Ascent with Approximate Projections 82

5.3 Learning Algorithms and Regret Bounds 87

5.3.1 Loop Free Episodic MDPCRs with Instructive Feedback 90

5.3.2 Loop Free Episodic MDPCRs with Evaluative Feedback 93

5.3.3 Uniformly Ergodic MDPCRs with Instructive Feedback 96

6 Conclusion **99**

Chapter 1

Introduction

This thesis focuses on a central goal of artificial intelligence: building computer systems capable of performing tasks or solving problems without the need for us, their designers, to treat each task or problem individually. That is, we want algorithms that enable computers to learn for themselves how to succeed at tasks and how to solve problems. I believe that a good approach is to first focus on designing algorithms for formal learning problems that we can mathematically reason about. Once we have a repertoire of formal problems and well-understood algorithms, we can then solve a real-world problem by first translating it into one of our formal learning problems and then applying an algorithm designed for that formal problem. If our formal model accurately captures the nature of the real-world problem, then the mathematical arguments we make about our learning algorithms will (nearly) hold in the real-world as well. I am not alone in this belief, and this strategy is common in the artificial intelligence community.

To create truly general learning algorithms we should also automate the modeling step, in which the real-world problem is approximated by a formal one. This is a very exciting and interesting research problem, but it appears to be quite difficult to make progress. Fortunately, even without automatic modeling, it is still worthwhile to study and design algorithms for formal learning problems. This is because it may be easier for a human to model a problem than to solve it. A computing scientist equipped with a set

of descriptive formal learning problems and good learning algorithms can then approach a difficult real-world problem by first modeling it formally and then handing it off to a computer. Moreover, when we do eventually have strategies for automatic modeling, it will be convenient to already have algorithms for many formal learning problems.

This thesis describes two projects in pursuit of the strategy outlined above. Both projects work towards answering interesting mathematical questions arising in the design and analysis of algorithms for two different formal learning problems. Both learning problems are formulated in the language of reinforcement learning, which is an approach whereby the computer learns by trial and error. Further, the algorithms studied in both projects treat learning as mathematical optimization and are derived from an optimization algorithm called gradient ascent. Finally, both projects measure learning performance in terms of regret, which is roughly how much worse the computer learner performed than if it had known the best strategy before hand. The title of the thesis, *Policy Gradient Reinforcement Learning Without Regret*, mentions explicitly each of these three components, which will be described in more detail in the remainder of the thesis.

The goal of the first project is to answer an open question about a family of algorithms called policy gradient methods. The question is somewhat technical, so for now I will only discuss it at a high level and postpone the detailed description until Chapter 4. In the reinforcement learning framework, the computer system receives a reward following each of its decisions and its goal is to earn the most reward in the long run. Policy gradient methods learn to earn rewards by trial and error. After trying one behaviour for a period of time, the algorithm compares the rewards it earned to a given baseline. If the computer performed better than the baseline, then the tendency to follow that behaviour again in the future is increased. Similarly, if the computer performed worse than the baseline, the likelihood of that behaviour is decreased. Surprisingly, the asymptotic formal guarantees for policy gradient methods do not depend on what baseline they compare against. The baseline does, however, influence the computer's finite-time behaviour, which leads immediately to the question: *what baseline should*

we use? This is the question addressed by the first project in this thesis. The answer to this question depends on our goals for the computer system. For example, some baselines may be computationally efficient to construct, while others may allow the computer to learn more quickly.

The main contributions of my first project are as follows: for two specific policy gradient algorithms, I propose that the baseline should be chosen to minimize the mean-squared-error of an internally used gradient estimate. I support this proposal by showing that under certain conditions, this choice results in a good regret bound for the two policy gradient algorithms. I present closed-form expressions showing how this baseline can be computed from properties of the environment. This closed form expression depends on properties of the environment that are usually unknown, so I also provide two new algorithms for estimating the baseline from interaction with the environment. Finally, I present an empirical comparison between three baselines that shows an empirical benefit to using my proposed baseline.

The second project combines aspects of reinforcement learning with aspects of another sub-field of computing science sometimes called online learning. The class of learning problems that are most commonly considered in reinforcement learning are called Markov decision processes, which are stochastic models describing how the computer's environment behaves. In contrast, problems from online learning typically use adversarial models, where we imagine that the environment is playing a game against the computer. In this project, we consider a class of learning problems called Markov decision processes with changing rewards (MDPCR), which are very similar to Markov decision processes where some stochastic pieces of the model are replaced with adversarial counterparts. The goal of this project is to design new efficient learning algorithms for this class of problems.

The main contributions of the second project, which were also presented at ICML 2014 [DGS2014], are as follows: I show that learning in Loop-free MDPCRs and Uniformly Ergodic MDPCRs can be reduced to another problem called online linear optimization. From this reduction, I derive three new learning algorithms with sublinear regret bounds. There are trivial algorithms that achieve regret bounds of the same order, but their compu-

tational complexity is exponential in the problem size. The three proposed algorithms all have computational complexity that is polynomial in the problem size. Moreover, for the so-called bandit information setting, the regret bound for the proposed algorithm holds under significantly weaker assumptions on the environment than existing algorithms. The three proposed algorithms are of interest because of their low complexity and since their analysis holds under weak conditions.

This thesis is organized as follows: (i) Chapter 2 introduces the reinforcement learning framework, Markov decision processes, and Markov decision processes with changing rewards. These are formal learning problems considered in this thesis; (ii) Chapter 3 introduces stochastic gradient descent and mirror descent, which are the optimization tools that are used in all algorithms studied in this thesis; (iii) Chapter 4 presents the first project, which focuses on the baseline for policy gradient methods; (iv) Chapter 5 presents the second project, which focuses on designing new algorithms for Markov decision processes with changing rewards; (v) and finally, Chapter 6 discusses directions for future research and gives concluding remarks.

Chapter 2

Reinforcement Learning and Decision Processes

This chapter introduces the reinforcement learning framework, Markov decision processes, and Markov decision processes with changing rewards. Both projects in this thesis work towards designing algorithms that effectively learn how to make decisions in one or the other of these two decision problems. The first project focuses on Markov decision processes, and the second project focuses on Markov decision processes with changing rewards.

The field of reinforcement learning is concerned with the following situation: A computer program, called the agent, is trying to achieve a well-specified goal while interacting with its environment. For example, an agent maintaining the inventory of a convenience store might be responsible for choosing how much of each product to order at the end of each week. In this setting, a natural goal for the agent is to maximize profits. If the agent orders too little of a popular product, then profits may be lost due to missed sales if it sells out. On the other hand, if the agent orders too much, profits may be lost if the excess items expire before being sold. To perform well at this task, the agent must interact with and anticipate the external world.

Every reinforcement learning problem has three components: states, actions, and rewards. In the above example, at the end of each week the agent chooses an action (the amount of each product to order) based on the

environment's current state (the store's current inventory and any other observations available to the agent). Following each action, the agent receives a scalar reward (the weekly profit), and the agent's goal is to maximize some measure of the long-term reward, such as the total profit over a fixed number of weeks, or the average profit per week.

Reinforcement learning problems differ in the set of states, the set of actions, and how the environment responds to the agent's actions. Markov decision processes and Markov decision processes with changing rewards are formal models for how the agent's actions affect the environment's state and rewards. We can mathematically reason about these formal models to make strong theoretical claims about learning algorithms. The two models presented in this chapter are complementary and each can be used to accurately model different kinds of real-world problems.

2.1 Markov decision processes

This section briefly describes Markov decision processes. The presentation below is heavily influenced by my interactions with Rich Sutton, András György, and Csaba Szepesvári, and the excellent books by Rich Sutton and Andy Barto [SB1998] and by Csaba Szepesvári [Cs2010].

Markov decision processes are *stochastic* models in that they suppose there are probability distributions that describe the outcomes of the agent's actions. For any set S, let Δ_S denote the set of probability distributions over S.[1] With this, we have the following definition:

Definition 2.1. A Markov decision process (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, x_{\text{start}}, \mathbb{T})$ where $\mathcal{X}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $x_{\text{start}} \in \mathcal{X}$ is the starting-state, and $\mathbb{T} : \mathcal{X} \times \mathcal{A} \to \Delta_{\mathcal{X}\times\mathbb{R}}$ is a transition probability kernel.

The interpretation of these quantities is as follows: Prior to choosing an

[1] When S is finite, we will consider probability measures with respect to the discrete σ-algebra. When S is the real line, we consider probability measures with respect to the Borel σ-algebra. Otherwise, S will be a product of finite sets and one or more copies of the real line, in which case we consider the product σ-algebra. The rest of this thesis does not discuss measure theoretic results.

action (encoded as an element $a \in \mathcal{A}$), the agent observes the environment's state (encoded as an element $x \in \mathcal{X}$). For each state action pair (x, a), the transition probability kernel gives a distribution over states and rewards, denoted by $\mathbb{T}(x, a)$. The environment's next state and the agent's reward are jointly distributed according to $\mathbb{T}(x, a)$ whenever the agent takes action a from state x. When the agent begins interacting with the environment, the environment is in state x_{start}. Typically, we consider the case when the agent knows the sets of states, the set of actions, and the starting state, but does not know the transition probability kernel.

We rarely need to work with the transition probability kernel directly. For almost all purposes, given a state-action pair (x, a), we only care about the marginal distribution over the next state and the expected reward. Therefore, we use the following simpler notation: Let (x, a) be any state-action pair and let (X', R) be randomly sampled from $\mathbb{T}(x, a)$. We define the *state transition probabilities* by $\mathcal{P}(x, a, x') = \mathbb{P}(X' = x')$ and the *expected reward* by $r(x, a) = \mathbb{E}[R]$. The dependence of these functions on the pair (x, a) is through the distribution of X' and R.

Just as it is useful to have a model for how the environment behaves, it is useful to have a model for how the agent chooses actions. For MDPs, a natural choice is to suppose that the agent chooses actions according to a Markov policy, which is a stochastic mapping from states to actions.

Definition 2.2. A (Markov) policy is a map $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$ that assigns a probability distribution over actions to each state. Let $\Pi = \Pi(\mathcal{X}, \mathcal{A})$ denote the set of all Markov policies.

We say that an agent is following policy π if, whenever the environment is in state x, she randomly chooses an action according to the distribution $\pi(x)$. We will denote by $\pi(x, a)$ the probability of choosing action a from state x.

Following any fixed policy $\pi \in \Pi$ will produce a random trajectory of states, actions, and rewards. The distribution on this trajectory depends only on the policy π and the MDP transition probability kernel $\mathbb{T}$. We will denote a sample of this random trajectory by X_1^π, A_1^π, R_1^π, X_2^π, A_2^π, R_2^π, ...

The meaning of the time indexes is as follows: action A_t^π was taken after observing X_t^π and the reward produced by this action was R_t^π. [2]

At first glance it seems restrictive that Markov policies are only permitted to choose actions based on the environment's current state, rather than the entire history of states, actions, and rewards. It turns out, however, that in all the cases we consider, there is an optimal policy that chooses actions as a deterministic function of the environment's current state. This is because, conditioned on the current state, the future of an MDP is independent of the history. We consider the more general class of (stochastic) Markov policies because they allow the agent to randomly choose actions, which is useful for trial and error learning.

Reinforcement learning algorithms adapt their behaviour over time to maximize reward. In principle, if the agent knew the environment's transition probability kernel $\mathbb{T}$ before hand, the agent could compute an optimal policy off-line prior to interacting with the environment. But, since the probability kernel is unknown, the agent must use its interaction with the environment to improve its policy. For example, a simple approach would be to compute a maximum likelihood estimate of the transition probability kernel based on the observed state transitions and rewards and to calculate the optimal policy for the approximated kernel. In general, this approach is too costly in terms of memory, computation, and interactions with the environment, so we seek other approaches.

The only remaining aspect of an MDP left to formalize is the agent's learning objective. That is, what exactly is the agent trying to accomplish while interacting with her environment? A formal learning objective is a map $J : \Pi \to \mathbb{R}$ that maps each policy to a scalar measure of its performance. The map J specifies precisely what we desire in a policy and is usually a function of both the policy and the environment. Intuitively, the learning objective should be to maximize some measure of the long-term reward the agent receives. The two most commonly used learning objectives are: First, maximizing the agent's expected total reward in repeated attempts at a

[2] This is somewhat non-standard, and often R_t^π is taken to be the reward produced by executing action A_{t-1}^π.

task. Second, maximizing the long-term average reward per-action in a task that continues indefinitely. The following subsections introduce these two learning objectives, together with additional conditions on the environment that make learning possible.

2.1.1 Total Reward in Episodic MDPs

The first formal learning objective that we consider is appropriate when the agent repeatedly tries the same task, and each attempt takes finite time. Each attempt is called an episode, and in this case, a natural formal goal is for the agent to maximize the total reward she earns in each episode. This objective is not appropriate when the agent's experience isn't naturally divided into episodes, since the total reward over an infinite span of time is generally also infinite. To accommodate this formal objective, we need to introduce the notion of episodes into the MDP model.

Definition 2.3. An MDP $(\mathcal{X}, \mathcal{A}, x_{\text{start}}, \mathbb{T})$ is said to be *episodic* if there exists a unique state $x_{\text{term}} \in \mathcal{X}$, called the terminal state, such that for all actions $a \in \mathcal{A}$ the transition kernel $\mathbb{T}(x_{\text{term}}, a)$ places all of its mass on $(x_{\text{term}}, 0)$. In other words, once the MDP enters state x_{term}, it remains there indefinitely while producing no reward.

Since nothing interesting happens after an episodic MDP enters its terminal state, we are free to restart the MDP and let the agent try again. We could also model the restarts directly in the MDP by adding a transition from the terminal state back to the starting state, but it is formally more convenient to have a single infinite trajectory of states, actions, and rewards for each episode (even if after some time it remains in the same state with zero reward).

The total episodic reward learning objective is defined as follows:

Definition 2.4. Let $\mathcal{M}$ be an episodic MDP. The expected total reward is a map $J_{\text{total}} : \Pi \to \mathbb{R}$ given by

$$J_{\text{total}}(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} R_t^\pi\right].$$

Let $\pi \in \Pi$ be any policy for an episodic MDP and let

$$T^\pi = \inf \{t \in \mathbb{N} : X_t^\pi = x_{\text{term}}\}$$

be the (random) first time that an agent following policy π enters the terminal state. Then we can rewrite the expected total reward as

$$J_{\text{total}}(\pi) = \mathbb{E}\left[\sum_{t=1}^{T^\pi - 1} R_t^\pi\right],$$

since after time T^π the rewards are 0 with probability one.

Two useful theoretical tools for learning to maximize total reward in episodic MDPs are the value and action-value functions. The value function measures the expected total reward an agent following policy π will receive starting from a given state x, and the action-value function measures the same when the agent starts from state x and takes action a first.

Definition 2.5. Let $\mathcal{M}$ be an episodic MDP. The value function $V_{\text{total}} : \mathcal{X} \times \Pi \to \mathbb{R}$ is defined by

$$V_{\text{total}}(x, \pi) = \mathbb{E}_x\left[\sum_{t=1}^{\infty} R_t^\pi\right],$$

where $\mathbb{E}_x$ denotes the expectation where the environment starts in state x, rather than x_{start}. For each policy π, the map $x \mapsto V_{\text{total}}(x, \pi)$ is usually called the value function for policy π.

Definition 2.6. Let $\mathcal{M}$ be an episodic MDP. The action-value function $Q_{\text{total}} : \mathcal{X} \times \mathcal{A} \times \Pi \to \mathbb{R}$ is defined by

$$Q_{\text{total}}(x, a, \pi) = \mathbb{E}_{x,a}\left[\sum_{t=1}^{\infty} R_t^\pi\right],$$

where $\mathbb{E}_{x,a}$ denotes the expectation where the environment starts in state x and the agent's first action is a. For each policy π, the map $(x, a) \mapsto Q_{\text{total}}(x, a, \pi)$ is usually called the action-value function for policy π.

Since the state transitions in an MDP do not depend on the history of states, any time an agent following policy π finds herself in state x, her expected total reward until the end of the episode is given by $V_{\text{total}}(x, \pi)$. Similarly, whenever the agent finds herself in state x and she takes action a, then $Q_{\text{total}}(x, a, \pi)$ is her expected total reward until the end of the episode. In other words, for any time t, we have

$$\mathbb{E}\left[\sum_{s=t}^{\infty} R_s^\pi \middle| X_t^\pi = x\right] = V_{\text{total}}(x, \pi)$$

and

$$\mathbb{E}\left[\sum_{s=t}^{\infty} R_s^\pi \middle| X_t^\pi = x, A_t^\pi = a\right] = Q_{\text{total}}(x, a, \pi)$$

whenever the events being conditioned on happen with non-zero probability.

2.1.2 Average Reward in Ergodic MDPs

The second formal learning objective that we consider is appropriate when we can't naturally divide the agent's experience into episodes. In this case, the agent's total reward on her infinite trajectory of states, actions, and rewards is generally also infinite. Given two policies that both have total reward diverging to infinity, how should we choose between them? A natural idea is to choose the policy that gives the fastest divergence. The long-term average reward per-action measures the asymptotic rate that a policy's total reward diverges.

Definition 2.7. Let $\mathcal{M}$ be any MDP. The average reward learning objective is a map $J_{\text{avg}} : \Pi \to \mathbb{R}$ defined by

$$J_{\text{avg}}(\pi) = \lim_{T \to \infty} \mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} R_t^\pi\right].$$

There is a potential problem with choosing policies that maximize J_{avg}: Since the agent changes her policy during her interaction with the environment, all of the policies she follows except for the first will *not* be started

from the starting state x_{start}. Therefore, if use J_{avg} to choose between policies, we would like to impose some constraints on the MDP that ensure the long-term average reward of a policy does not depend on the starting state. Otherwise, the agent may be encouraged to choose a policy which has high average reward when started from the starting state, but which performs poorly given the environment's current state.

A relatively mild condition on the environment that ensures the average reward of a policy does not depend on the starting state is ergodicity:

Definition 2.8. An MDP $(\mathcal{X}, \mathcal{A}, x_{\text{start}}, \mathbb{T})$ is said to be *weakly ergodic*[3] if, for each policy $\pi \in \Pi$, there exists a unique distribution $\nu(\pi) \in \Delta_{\mathcal{X}}$, called the stationary distribution of π, such that

$$\nu(x, \pi) = \sum_{a \in \mathcal{A}} \pi(x, a) \sum_{x' \in \mathcal{X}} \mathcal{P}(x, a, x') \nu(x', \pi),$$

where $\nu(x, \pi)$ denotes the probability mass given to state x by the distribution $\nu(\pi)$.

The condition in Definition 2.8 is a fixed-point equation and states that if I sample a random state x from $\nu(\pi)$ and then take a single step according to the policy π to get a new state x', then the distribution of x' is exactly $\nu(\pi)$ again.

It is well known that in weakly ergodic MDPs, it is possible to rewrite the average reward learning objective in terms of the stationary distribution as follows:

$$\begin{aligned} J_{\text{avg}}(\pi) &= \sum_{x,a} \nu(x, \pi) \pi(x, a) r(x, a) \\ &= \mathbb{E}[r(X, A)], \quad \text{where} \quad X \sim \nu(\pi), A \sim \pi(X). \end{aligned}$$

The starting state of the MDP no longer appears in this expression for the average reward, and therefore the average reward does not depend on the starting state.

[3] In the remainder of this thesis, I will refer to weakly ergodic MDPs simply as ergodic MDPs.

Like in the total reward setting, the value and action-value functions are useful theoretical tools which have essentially the same interpretation as before. Now, rather than measuring the total reward following a given state, they measure the transient benefit of being in a give state, or state-action pair, compared with the long-term average.

Definition 2.9. Let $\mathcal{M}$ be an ergodic MDP. The value function $V_{\text{avg}} : \mathcal{X} \times \Pi \to \mathbb{R}$ is defined by

$$V_{\text{avg}}(x, \pi) = \mathbb{E}_x \left[\sum_{t=1}^{\infty} \big(R_t^\pi - J_{\text{avg}}(\pi) \big) \right].$$

Definition 2.10. Let $\mathcal{M}$ be an ergodic MDP. The action-value function $Q_{\text{avg}} : \mathcal{X} \times \Pi \to \mathbb{R}$ is defined by

$$Q_{\text{avg}}(x, a, \pi) = \mathbb{E}_{x,a} \left[\sum_{t=1}^{\infty} \big(R_t^\pi - J_{\text{avg}}(\pi) \big) \right].$$

2.2 Markov deicison processes with changing rewards

MDPs are not suitable models for all environments. In particular, since the transition probability kernel $\mathbb{T}$ of an MDP is unchanging with time, it can be difficult to model environments whose dynamics change over time. This section describes Markov decision processes with changing rewards (MDPCRs), which are a class of environment model that capture some kinds of non-stationarity. This class of problems has been the focus of recent research efforts [EKM2005, EKM2009, NGS2010, NGSA2014, YMS2009] and goes by several different names, the most common of which is "Online MDP". I choose to use the name MDPCR because "Online MDP" is not universally used, and I think MDPCR is more descriptive.

Before describing MDPCRs, I would like to comment on an alternative approach to modeling non-stationary environments. In principle, we can model a non-stationary environments as an MDP by including a description

of the environment's current behaviour in the state. For example, if the environment switches between a finite number of modes, then we could include an integer in the MDP state that indicates which mode the environment is currently in. The drawback of this approach is that, in the MDP framework, the agent completely observes the state, so the agent must be able to observe the environment's current mode, which is a rather strong requirement. One way to avoid this requirement is to modify the MDP model so that the agent only partially observes the state. This kind of model is called a partially observable MDP (POMDP). POMDPs are an interesting research topic, but not a focus of this thesis.

MDPCRs are a different approach to modeling non-stationary environments. They keep the assumption that the agent completely observes the modeled state and that the state transitions are Markovian. The difference is that the reward for taking action a from state x changes over time in a non-stochastic way. Specifically, there is an unknown sequence of reward functions $r_1, r_2, \ldots$ and executing action a_t from state x_t at time t produces a reward $r_t(x_t, a_t)$.

Definition 2.11. A Markov decision process with changing rewards (MDPCR) is a tuple $\left(\mathcal{X}, \mathcal{A}, x_{\text{start}}, \mathcal{P}, (r_t)_{t\in\mathbb{N}}\right)$ where $\mathcal{X}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $x_{\text{start}} \in \mathcal{X}$ is the starting-state, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \Delta_{\mathcal{X}}$ encodes the state transition probabilities, and each $r_t : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a reward function.

The interpretations of all quantities in Definition 2.11 is the same as for regular MDPs with the exception of the sequence of reward functions. In this thesis, I consider the case where the agent knows the set of states, the set of actions, the starting state, and the state transition probabilities, and the only unknown quantity is the sequence of reward functions.

There are two different protocols under which the agent learns about the reward functions. The first, which is sometimes called evaluative feedback (or bandit feedback), is where the agent only observes the scalar value $r_t(X_t, A_t)$ after executing action A_t from state X_t. The second, which is sometimes called instructive feedback (or full-information feedback) is where

the agent observes the entire reward function $r_t : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ after choosing an action at time t. The evaluative feedback setting is more useful in real-world tasks, since often the rewards are determined by some real-world process and we only see the outcome of the action that was taken. The instructive feedback case is still interesting theoretically, sometimes useful in practice, and acts as a stepping stone towards developing algorithms for the evaluative feedback setting.

The main usefulness of MDPCRs is modeling tasks where the agent's rewards depend on a difficult-to-model aspect of the environment. For example, suppose the agent's task is to explore a maze searching for treasure. The agent's actions have predictable (and time-invariant) effects on her location in the maze and are therefore easily modeled with Markovian transition probabilities. Suppose, however, that there is a wizard that periodically creates and destroys treasures throughout the maze. The agent's reward depends not only on her position in the maze, but also on the recent actions of the wizard. This problem is difficult to model as an MDP, since the rewards must be sampled from a time-invariant distribution that depends only on the most recent state and action. This forces the state to explicitly model (at least part of) the wizard's behaviour, which may be very complicated. On the other hand, it is easy to model this task as an MDPCR, since we can leave the wizard out of the state entirely and use the sequence of reward functions to model the moving treasures. Similar problems arise in many situations where the agent interacts with another entity with agency, such as a human user or another machine.

Like in the MDP case, we consider two different formal learning objectives: one suitable for episodic tasks and one suitable for continuing tasks. For each formal learning objective, we consider a sub-class of MDPCRs where learning is possible. The sub-classes considered in this thesis are more restrictive than in the MDP setting, and an interesting open question is to design learning algorithms for more general settings. In the episodic setting, we consider learning in *loop-free episodic MDPCRs* and in the continuing setting, we consider *uniformly ergodic MDPCRs*.

Before giving detailed descriptions of the two cases, I will discuss some

common features of both models.

In each case, we define a sequence of performance functions J_1, J_2, ..., where $J_T(\pi_1, \ldots, \pi_T)$ is a function of T policies and represents the expected performance of an agent following policies π_1, ..., π_T for the first T time-steps. For example, we might define

$$J_T(\pi_1, \ldots, \pi_T) = \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right]$$

to be the expected total reward earned by an agent following policies π_1, ..., π_T for the first T time steps. The reason why we need a sequence of performance functions, rather than a single performance function like for MDPs, is because the performance depends on the changing sequence of reward functions. This thesis focuses on the formal learning objective of maximizing $J_T(\pi_1, \ldots, \pi_T)$ for some fixed time-horizon T. There are standard techniques, such as the doubling trick (see, for example, Section 2.3.1 of [S2012]) that allow these algorithms to be extended to the case when the time-horizon T is not known in advance.

In our formal analysis, it will be more convenient to work with the *regret*, which is defined below.

Definition 2.12. Let J_1, J_2, ... be any sequence of performance functions. The regret of the sequence of policies $\pi_1, \ldots, \pi_T \in \Pi$ relative to a fixed policy $\pi \in \Pi$ at time (or episode) T is given by

$$\mathcal{R}_T(\pi_1, \ldots, \pi_T; \pi) = J_T(\pi, \ldots, \pi) - J_T(\pi_1, \ldots, \pi_T).$$

In words, it is the gap in performance between an agent that follows policies $\pi_1, \ldots, \pi_T$ and an agent that follows policy π on every time step. The regret of the sequence relative to the set of Markov policies is given by

$$\begin{aligned}\mathcal{R}_T(\pi_1, \ldots, \pi_T) &= \sup_{\pi \in \Pi} \mathcal{R}_T(\pi_1, \ldots, \pi_T; \pi) \\ &= \big(\sup_{\pi \in \Pi} J_t(\pi, \ldots, \pi)\big) - J_T(\pi_1, \ldots, \pi_T).\end{aligned}$$

Minimizing regret is equivalent to maximizing the performance function, but as we will see later, the regret is easier to analyze. In particular, we will be able to provide upper bounds on the regret which depend only loosely on the actual sequence of rewards.

2.2.1 Loop-free Episodic MDPCRs

A loop-free episodic MDPCR is much like an episodic MDP with two additional constraints: The agent can never visit the same state twice in a single episode and every episode has the same length. Formally, we have the following definition

Definition 2.13. A *loop-free MDPCR* is a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, (r_t)_{t\in\mathbb{N}})$ such that the state space $\mathcal{X}$ can be partitioned into L layers $\mathcal{X}_1, \ldots, \mathcal{X}_L$ with the following properties:

1. the first layer contains a unique starting state: $\mathcal{X}_1 = \{x_{\text{start}}\}$;
2. the last layer contains a unique terminal state: $\mathcal{X}_L = \{x_{\text{term}}\}$;
3. for every action a, we have $r_L(x_{\text{term}}, a) = 0$;
4. and, for any states x and x' and any action a, if $\mathcal{P}(x, a, x') > 0$, then either $x = x' = x_{\text{term}}$ or there exists a layer index $1 \leq l < L$ such that $x \in \mathcal{X}_l$ and $x' \in \mathcal{X}_{l+1}$.

The above conditions guarantee that every episode in a loop-free episodic MDPAC will visit exactly L distinct states, one from each layer. The agent starts in the first layer, which contains only the starting state, and proceeds through the layers until arriving at the final layer, which contains only the terminal state. Once the agent enters the terminal state, the rest of the trajectory remains in the terminal state and produces zero reward.

It is natural to measure time in loop-free episodic MDPCRs by counting episodes, rather than time steps. Since the agent will never return to any state in a single episode, there is no reason for her to update her action selection probabilities for that state before the end of the episode. Similarly,

we can take the duration of each reward function to be an entire episode, rather than a single time step. Therefore, we denote by π_τ and r_τ the agent's policy and the reward function for episode τ, respectively. Finally, we measure the agent's regret after T episodes, rather than after T time steps of interaction.

The performance function that we consider in loop-free episodic MDPCRs is the total reward earned over the first T episodes:

Definition 2.14. In an MDPCR, the expected total reward of the policies $\pi_1, \ldots, \pi_T$ in the first T episodes is given by

$$J_{\text{total},T}(\pi_1, \ldots, \pi_T) = \sum_{\tau=1}^{T} \mathbb{E}_{\pi_\tau}\left[\sum_{t=1}^{L} r_\tau(X_t, A_t)\right],$$

where $\mathbb{E}_{\pi_\tau}$ denotes the expectation where actions are selected according to policy π_τ.

For this performance function, we can write the regret (relative to all Markov policies) as follows:

$$\mathcal{R}_{\text{total},T}(\pi_1, \ldots, \pi_T) = \sup_{\pi \in \Pi} \sum_{\tau=1}^{T} \left\{ \mathbb{E}_\pi\left[\sum_{t=1}^{L} r_t(X_t, A_t)\right] - \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{L} r_t(X_t, A_t)\right] \right\}.$$

Suppose that an agent's regret grows sublinearly with T. Then for any policy π, we have

$$\begin{aligned}\mathcal{R}_{\text{total},T}(\pi_{1:T}; \pi)/T &= \frac{1}{T}\sum_{\tau=1}^{T} \mathbb{E}_\pi\left[\sum_{t=1}^{L} r_\tau(X_t, A_t)\right] - \frac{1}{T}\sum_{\tau=1}^{T} \mathbb{E}_{\pi_\tau}\left[\sum_{t=1}^{L} r_\tau(X_t, A_t)\right] \\ &\leq \mathcal{R}_{\text{total},T}(\pi_1, \ldots, \pi_T)/T \to 0.\end{aligned}$$

Taking π to be the best Markov policy, we have that the average episodic reward of the agent is converging to the average episodic reward of the best Markov policy. Therefore, our main goal is to show that the regret of our algorithms grows sublinearly. Naturally, slower growth rates are more desirable.

2.2.2 Uniformly Ergodic MDPCRs

Uniformly ergodic MDPCRs are very similar to ergodic MDPs. Recall that ergodic MDPs were characterized by the existence of a unique stationary distribution $\nu(\pi) \in \Delta_{\mathcal{X}}$ for each policy π that described the long-term average state visitation probabilities while following policy π. The condition in uniformly ergodic MDPs guarantees that every policy has a unique stationary distribution, that the finite-time state-visitation probabilities $\nu_t(\pi)$ ($\nu_t(x, \pi) = \mathbb{P}(X_t^\theta = x)$) converge to the unique stationary distribution, and that the rate of convergence of $\nu_t(\pi)$ to $\nu(\pi)$ is uniformly fast over all policies. Formally, we have the following definition

Definition 2.15. An MDPCR $\left(\mathcal{X}, \mathcal{A}, x_{\text{start}}, \mathcal{P}, (r_t)_{t\in\mathbb{N}}\right)$ is said to be *uniformly ergodic* if there exists a constant $\tau \geq 0$ such that for any two distributions ν and $\nu' \in \Delta_{\mathcal{X}}$ and any policy π, we have

$$\left\|\nu P^\pi - \nu' P^\pi\right\|_1 \leq e^{-1/\tau} \left\|\nu - \nu'\right\|_1,$$

where P^π is linear operator on distributions corresponding to taking a single step according to π, defined component-wise as follows:

$$(\nu P^\pi)(x') = \sum_x \nu(x) \sum_a \pi(x, a)\mathcal{P}(x, a, x').$$

The above condition implies ergodicity in the sense of Definition 2.8. Suppose an agent follows policy π on every time step and let $\nu_t(\pi) \in \Delta_{\mathcal{X}}$ denote her probability distribution over states at time t. Since the agent starts in state x_{start}, we have that $\nu_1(x, \pi) = \mathbb{I}\{x = x_{\text{start}}\}$. Using the notation from above, it is not difficult to check that $\nu_t(\pi) = \nu_1(P^\pi)^{(t-1)}$. The condition shows that the operator P^π is a contraction and, by the Banach fixed point theorem, we have that the sequence $\nu_t(\pi)$ converges to the unique fixed point of P^π, which we denote by $\nu(\pi)$. Being a fixed point of P^π is exactly the definition of a stationary distribution from Definition 2.8.

Uniform ergodicity is a considerably stronger requirement than ergodicity, and an interesting open question is to decide if there exist learning

algorithms for (non-uniform) ergodic MDPCRs with provably good performance.

The performance function we use for uniformly ergodic MDPCRs is very similar to the one used for the loop-free episodic MDPCRs, except that the total reward is measured in terms of time steps, rather than episodes:

Definition 2.16. In a uniformly ergodic MDPCR, the expected total reward of the policies $\pi_1, \ldots, \pi_T$ in the first T time steps is given by

$$J_{\text{total},T}(\pi_1, \ldots, \pi_T) = \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right],$$

where $\mathbb{E}_{\pi_{1:T}}$ denotes the expectation where the t^{th} action A_t is chosen according to policy π_t.

For this performance function, we can write the regret (relative to all Markov policies) as follows:

$$\mathcal{R}_{\text{total},T}(\pi_1, \ldots, \pi_T) = \sup_{\pi \in \Pi} \left\{ \mathbb{E}_{\pi}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] - \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] \right\}.$$

By exactly the same argument as for loop-free episodic MDPCRs, an agent with sublinear regret will have average reward converging to the average reward of the best Markov policy.

Chapter 3

Gradient Methods

This chapter introduces three optimization algorithms: *gradient ascent*, *stochastic gradient ascent*, and *online mirror ascent*. These algorithms are relevant to this thesis because all of the considered learning algorithms are based on mathematical optimization. Policy gradient methods, which are the focus of the first project, are an instance of stochastic gradient ascent. All three algorithms introduced in the second project are instances of online mirror ascent. Gradient ascent is included in the discussion because it is a good starting point for describing the other two.

The goal of mathematical optimization can be stated formally as follows: Given a function $f : K \to \mathbb{R}$ where $K \subset \mathbb{R}^d$ is a subset of d-dimensional space, find a vector $w \in K$ that maximizes $f(w)$. We use the following notation to write this problem:

$$\operatorname*{argmax}_{w \in K} f(w).$$

The difficulty of finding a maximizer depends heavily on the structural properties of f and K. For example, when f is a concave function and K is a convex set, the global maximizer of f can be efficiently found. When f is not concave, the best we can hope for is to find a local maximum of the function f.

3.1 Gradient Ascent

The gradient ascent algorithm can be applied whenever the objective function f is differentiable. Gradient ascent produces a sequence of vectors w_1, w_2, ... such that the function value of f along the sequence is increasing. In this section, we only consider so-called unconstrained maximization problems where $K = \mathbb{R}^d$. Pseudocode for gradient ascent is given in Algorithm 1.

Input: step-size $\eta > 0$.
Choose $w_1 = 0 \in \mathbb{R}^d$;
for *each time* $t = 1, 2, \ldots$ **do**
    Optionally use w_t in some other computation;
    Set $w_{t+1} = w_t + \eta \nabla f(w_t)$;
end

Algorithm 1: Gradient Ascent

There is a simple geometric idea underlying gradient asecnt. We imagine that the graph of the function f is a landscape, where $f(w)$ is the height at location w (this analogy works best in $\mathbb{R}^2$). The gradient $\nabla f(w)$ is a vector that points in the direction from w that f increases the most rapidly. Therefore, the gradient ascent update $w_{t+1} = w_t + \eta \nabla f(w_t)$ produces w_{t+1} by taking a small step up-hill from w_t. It is appropriate to think of gradient ascent as optimizing a function as though it were a walker searching for the highest point in a park by walking up hill.

We can also motivate the gradient ascent update as maximizing a linear approximation to the function f. Specifically, since f is differentiable, the first order Taylor expansion gives a linear approximation to f:

$$f(u) \approx f(w) + \nabla f(w)^\top (u - w).$$

This approximation is accurate whenever u is sufficiently close to w. A naive idea would be to fix some $w_0 \in \mathbb{R}^d$ and maximize the linear approximation $w \mapsto f(w_0) + \nabla f(w_0)^\top (w - w_0)$. There are two problems with this approach: first, the linearized objective function is unbounded (unless it is constant)

and therefore has no maximizer. Second, the linear objective is only a good approximation of f near the point w_0, so we should only trust solutions that are near to w_0. A better idea is to successively maximize linear approximations to the function f, with a penalty that prevents our solutions from being too far from the region where the approximation is accurate. Formally, we might set

$$w_{t+1} = \operatorname*{argmax}_{u \in \mathbb{R}^d} \eta\big(f(w_t) + \nabla f(w_t)^\top (u - w_t)\big) - \frac{1}{2} \|u - w_t\|_2^2 .$$

The objective function in the above update has two competing terms. The first term encourages w_{t+1} to maximize the linear approximation of f at the point w_t and the second term encourages w_{t+1} to stay near to the point w_t. The step size parameter η trades off between these two competing objectives. The above objective is a concave quadratic function of u, and we can express its unique maximizer in closed form as

$$\begin{aligned} w_{t+1} &= \operatorname*{argmax}_{u \in \mathbb{R}^d} \eta\big(f(w_t) + \nabla f(w_t)^\top (u - w_t)\big) - \frac{1}{2} \|u - w_t\|_2^2 \\ &= w_t + \eta \nabla f(w_t), \end{aligned}$$

which is exactly the gradient ascent update. Therefore, we may think of the gradient ascent update rule as maximizing a linear approximation to the function f with an additional penalty that keeps the maximizer near to the previous guess. When we view gradient ascent in this way, we see that the update equation is defined based on the squared 2-norm. We will see in Section 3.3 that we can derive similar algorithms where the the squared 2-norm distance is replaced by another distance function.

In addition to the above intuitions, we care that gradient ascent actually maximizes functions. The following theorem guarantees that as long as the step size is sufficiently small, the gradient ascent algorithm converges to a stationary point of the function f.

Theorem 3.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be such that $f^* = \sup_w f(w) < \infty$ and ∇f*

is L-Lipschitz. That is,

$$\text{for all } u, v \in \mathbb{R}^d \quad \|\nabla f(u) - \nabla f(v)\|_2 \leq \|u - v\|_2 .$$

If the step-size satisfies $\eta < 1/L$, then the sequence $(w_t)_t$ produced by gradient ascent converges to some point w_∞ such that $\nabla f(w_\infty) = 0$.

In most cases, this result is enough to guarantee that gradient ascent converges to a local maximum of the function f. However, it is possible to get unlucky and converge to some other point where the gradient of f is zero, such as a local minima or a saddle-point of f. Since maximizing an arbitrary non-concave functions is computationally intractible, this is the best result we can hope for.

When the function f is concave, the situation is much better:

Theorem 3.2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a concave function with maximizer w^*. Let $(w_t)_t$ be the sequence of points produced by running gradient ascent with step size $\eta > 0$. Suppose that $\|w^*\|_2 \leq B$ and $\|\nabla f(w_t)\|_2 \leq G$ for all times $t = 1, \ldots, T$. Then*

$$\sum_{t=1}^{T} \big(f(w^*) - f(w_t)\big) \leq \frac{B^2}{\eta} + \eta T G^2.$$

Setting the step size to be

$$\eta = \frac{B}{G\sqrt{T}}$$

gives the best bound of

$$\sum_{t=1}^{T} \big(f(w^*) - f(w_t)\big) \leq 2BG\sqrt{T}.$$

Proof. This is a standard result. Since f is concave, for any points u and w in $\mathbb{R}^d$, we have

$$f(w) \leq f(u) + \nabla f(u)^\top (w - u).$$

Rearranging this inequality gives $f(w) - f(u) \leq \nabla f(u)^\top (w - u)$. Taking $u = w_t$ and $w = w^*$ gives $f(w^*) - f(w_t) \leq \nabla f(w_t)^\top (w^* - w_t)$. Summing

over times t, we have

$$\sum_{t=1}^{T}(f(w^*) - f(w_t)) \leq \sum_{t=1}^{T} \nabla f(w_t)^\top (w^* - w_t).$$

Theorem 5.11 and Lemma 5.12 together with the facts that $\|\cdot\|_2$ is self-dual and 1-strongly convex with repsect to itself give the final result. The above theorem and lemma are the main subject of Section 5.2. □

This results shows that whenever we set the step size appropriately, the total suboptimality (usually called the regret) of the sequence $w_1, \ldots, w_T$ produced by gradient ascent grows at a rate of only $\sqrt{T}$. Equivalently, dividing by T shows that the average suboptimality $\frac{1}{T}\sum_{t=1}^{T}(f(w^*) - f(w_t))$ goes to zero at least as quickly as $1/\sqrt{T}$.

3.2 Stochastic Gradient Ascent

Stochastic gradient ascent is a variant of the gradient ascent algorithm that can sometimes be used to maximize a function f even if we can't compute the value of f or its gradient ∇f. This method requires only that we are able to produce random vectors whose expectation is equal to the gradient of the function f. The idea behind stochastic gradient ascent is simply to use these stochastic gradient estimates in place of the true gradients. Pseudocode is given in Algorithm 2.

Input: step-size $\eta > 0$.
Choose $w_1 = 0 \in \mathbb{R}^d$;
for *each time* $t = 1, 2, \ldots$ **do**
  Optionally use w_t in some other computation;
  Set $w_{t+1} = w_t + \eta \nabla_t$ where $\mathbb{E}[\nabla_t \,|\, w_t] = \nabla f(w_t)$;
end

Algorithm 2: Stochastic Gradient Ascent

One common situation where we can't evaluate f or ∇f, but for which

we can get unbiased stochastic estimates of the gradient is as follows: Let P be a probability distribution that is unknown, but from which we can sample. Set $f(w) = \mathbb{E}[g(w, X)]$ where g is a known function, and X has distribution P. Since the distribution of X is unknown, we can't evaluate f or its gradient ∇f. But, let X have distribution P and set $\nabla = \nabla_w g(w, X)$. Then we have

$$\mathbb{E}[\nabla] = \mathbb{E}[\nabla_w g(w, X)] = \nabla_w \mathbb{E}[g(w, X)] = \nabla f(w).$$

Therefore, even though we can't compute f or ∇f, we can produce a random vector whose expectation is equal to $\nabla f(w)$ for any $w \in \mathbb{R}^d$.

In Algorithm 2, the condition on ∇_t is that $\mathbb{E}[\nabla_t \,|\, w_t] = \nabla f(w_t)$. The reason that the conditional expectation is used instead of a plain expectation is that the sequence w_t is itself random. It will generally not be the case that the random vector $\nabla f(w_t)$ is equal to the constant $\mathbb{E}[\nabla_t]$. The condition $\mathbb{E}[\nabla_t \,|\, w_t] = \nabla f(w_t)$ is roughly equivalent to "given the value of w_t, the expectation of ∇_t should be equal to $\nabla f(w_t)$."

Since the sequence $(w_t)_{t \in \mathbb{N}}$ produced by Algorithm 2 is random, we can only make probabilistic statements about how well the sequence optimizes f. When the function f is non-concave, a standard results shows that the random sequence $(w_t)_t$ produced by stochastic gradient ascent almost surely converges to a local maxima of the function f when a time-varying step size is used that goes to zero at an appropriate rate. In practice, a constant step size is often used instead, since this results in faster convergence early in the optimization, at the cost of never quite driving the error to zero. I omit the exact details of these results since they are quite technical and never used directly in this thesis.

As with the deterministic case, the situation is much better when the function f is concave. In this case, following essentially the same approach as before, we have the following upper bound on the expected total suboptimality (regret) of stochastic gradient ascent.

Theorem 3.3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a concave function with maximizer w^*. Let $(w_t)_t$ be the sequence of points produced by running stochastic gradient*

ascent with step size $\eta > 0$ *and gradient estimates* $(\nabla_t)_t$. *Suppose that* $\|w^*\|_2 \leq B$ *and* $\mathbb{E}[\|\nabla f(w_t)\|_2^2] \leq G^2$ *for all times* $t = 1, \ldots, T$. *Then*

$$\mathbb{E}\Big[\sum_{t=1}^{T}\big(f(w^*) - f(w_t)\big)\Big] \leq \frac{B^2}{\eta} + \eta T G^2.$$

Setting the step size to be

$$\eta = \frac{B}{G\sqrt{T}}$$

gives the best bound of

$$\mathbb{E}\Big[\sum_{t=1}^{T}\big(f(w^*) - f(w_t)\big)\Big] \leq 2BG\sqrt{T}.$$

Proof. The proof of this result is essentially identical to the proof of Theorem 3.2 and is omitted. □

Notice that the bound in Theorem 3.3 only depends on the distribution of the gradient estimatse ∇_t by way of their second moment. Therefore, to get the best possible bound, one should try to construct the gradient estimates to have the smallest possible second moment.

3.3 Online Mirror Ascent

This section introduces online mirror ascent, which generalizes gradient ascent in two ways. First, it is an algorithm for a problem called online linear optimization, which is a slightly more general problem than maximizing a function f using only the gradient of f. Second, online mirror ascent has an additional parameter called the regularizer function $R : \mathbb{R}^d \to \mathbb{R}$ that defines a distance function that replaces the squared 2-norm distance. The regularizer allows mirror ascent to better exploit the natural geometry of an optimization problem. If we take the regularizer to be $R(w) = \frac{1}{2}\|w\|_2^2$, then we recover exactly gradient ascent, but there are other interesting cases as well. For example, if the vectors represent probability distributions, then we

may way to measure distances in terms of the Kullback-Leibler divergence instead of the squared 2-norm distance.

The problem solved by online mirror ascent is called online linear optimization, defined as follows:

Definition 3.4. Online linear optimization is a game played between an agent and her environment. On round t of the game, the agent chooses a point w_t from a convex set $K \subset \mathbb{R}^d$. Following the agent's choice, the environment chooses a payout vector $r_t \in \mathbb{R}^d$ and the agent earns reward given by the inner product $r_t^\top w_t$. The set K is fixed for all rounds of the game. The agent's choice w_t may only depend on $w_{1:(t-1)}$ and $r_{1:(t-1)}$, while the environment's choice of r_t may depend on $w_{1:t}$ and $r_{1:(t-1)}$.

Given a fixed time horizon T, the agent's goal is to maximize her total reward in the first T rounds. Equivalently, she can minimize her regret relative to the set K, given by

$$\mathcal{R}_T(w_{1:T}, r_{1:T}) = \sup_{w \in K} \sum_{t=1}^{T} r_t^\top w - \sum_{t=1}^{T} r_t^\top w_t = \sup_{w \in K} \sum_{t=1}^{T} r_t^\top (w - w_t).$$

We treat the online linear optimization problem in a game-theoretic style and prove bounds on the regret for the worst-case sequence of payout vectors $r_{1:T}$ under the constraint that $r_t(w) \in [0, 1]$ for all rounds t and $w \in K$.

The online mirror ascent algorithm is similar in spirit to gradient ascent, but different in three important ways. First, rather than using the gradient of a function to construct its update, it uses the payout vector from an online linear optimization game. Second, rather than having an update that is defined in terms of the squared euclidian distance (as in gradient ascent), the update is defined in terms of a so-called Bregman divergence, which allows the algorihtm to better take advantage of the underlying geometry of a problem. For example, if the set K consists of probability distributions, then it may make more sense to measure distance between them by the KL-divergence than by their squared Euclidian distance. Finally, we will present online mirror ascent for constrained online linear optimization, where K is a proper subset of $\mathbb{R}^d$. In principle, (stochastic) gradient ascent can

also accomodate constrained optimization problems, but this discussion was omitted above because it is not used in this thesis.

I will now introduce Bregman divergences and describe how they are used in the mirror ascent update. First, we need the notion of a strongly convex function:

Definition 3.5. A function $f : S \to \mathbb{R}$ with $S \subset \mathbb{R}^d$ is said to be σ-strongly convex with respect to the norm $\|\cdot\|$ if

$$f(u) \geq f(w) + \nabla f(w)^\top (u - w) + \frac{\sigma}{2} \|u - w\|^2$$

for all vectors u and w in S.

Each strongly convex function induces a Bregman divergence on the domain S, which is similar to a distance function.

Definition 3.6. Let $R : S \to \mathbb{R}$ be a σ-strongly convex function with respect to the norm $\|\cdot\|$. The Bregman divergence induced by R is a map $D_R : S \times S^\circ \to \mathbb{R}$ defined by

$$D_R(u, w) = R(u) - R(w) - \nabla R(w)^\top (u - w),$$

where S° denotes the interior of S.

The following lemma establishes some properties of Bregman divergences that show they are somewhat similar to distance functions:

Lemma 3.7. *Let $R : S \to \mathbb{R}^d$ be a σ-strongly convex function with respect to the norm $\|\cdot\|$ and D_R be the induced Bregman divergence. Then the following statements hold*

1. *$D_R(u, v) \geq 0$ for all vectors u and v in S,*
2. *$D_R(u, v) = 0$ if and only if $u = v$,*
3. *(Pythagorean Theorem) If K is a convex subset of S, $w \in S$, $u \in K$, and we set $w' = \operatorname{argmin}_{v \in K} D_R(v, w)$, then $D_R(u, w) \geq D_R(u, w') + D_R(w', w)$.*

Even though Bregman divergences behaving like distances, they are not usual distances because they are not symmetric and do not satisfy the triangle inequality.

With these definitions in hand, we are ready to define online mirror ascent. Algorithm 3 gives pseudocode. The online mirror ascent update has two steps: first, we compute an unconstrained maximizer $w_{t+1/2}$ of the most recent payout vector together with a penalty that encourages $w_{t+1/2}$ to not stray too far from w_t. Usually this update step has a closed-form expression that can be efficiently evaluated. The second step is to set w_{t+1} to be the projection of $w_{t+1/2}$ back onto the constraint set K with respect to the Bregman divergence D_R. Theorem 3.8 bounds the regret of online mirror ascent.

Theorem 3.8. *Let $R : S \to \mathbb{R}$ be a σ-strongly convex regularizer with respect to the norm $\|\cdot\|$ and $K \subset S$ be a convex set. Then for any sequence of payout vectors r_t and any fixed point $w \in K$, the regret (relative to w) of online mirror ascent with step size $\eta > 0$ and regularizer R satisfies*

$$\sum_{t=1}^{T} r_t^\top (w - w_t) \leq \frac{D_R(w, w_1)}{\eta} + \frac{\eta}{\sigma} \sum_{t=1}^{T} \|r_t\|_*^2 ,$$

where $\|\cdot\|_*$ is the dual norm of r_t. Moreover, if $\|r_t\|_* \leq G$ for all t, then we have

$$\begin{aligned} \sum_{t=1}^{T} r_t^\top (w - w_t) &\leq \frac{D_R(w, w_1)}{\eta} + \frac{\eta T G^2}{\sigma} \\ &= 2G\sqrt{T D_R(w, w_1)/\sigma} \end{aligned}$$

where the last line is obtained by taking eta $= \sqrt{\frac{\sigma D_R(w,w_1)}{TG^2}}$, which is the optimal value.

Proof. As before, this result is a special case of Theorem 5.11 together with Lemma 5.12, so I defer the proof until Section 5.2. □

Input: Step size $\eta > 0$, Regularizer $R : S \to \mathbb{R}$ with $S \supset K$

Choose $w_1 \in K$ arbitrarily;

for *Each round* $t = 1, 2, \ldots$ **do**

  Optionally use w_t in another computation;

  Set $w_{t+1/2} = \operatorname{argmax}_{u \in S} \eta r_t^\top u - D_R(u, w_t)$;

  Set $w_{t+1} = \operatorname{argmin}_{u \in K} D_R(u, w_{t+1/2})$;

end

Algorithm 3: Online mirror ascent

Chapter 4

Policy Gradient Methods and Baseline Functions

This chapter describes the first project that I worked on during my MSc program. The goal of this project was to resolve an open question related to policy gradient methods, which are learning algorithms for MDPs. Policy gradient methods are instances of stochatic gradient ascent. Recall that to apply stochastic gradient ascent, we must be able to sample random vectors whose expectation is equal to the gradient of the objective function. For both the total and average reward learning objectives, there are established theoretical results that give methods for generating random gradient estimates. Both gradient estimation schemes have a parameter called the baseline function. The baseline does not change the expectation of the gradient estimates and therefore has no influence on the asymptotic performance of policy gradient methods. The baseline function does, however, impact the finite-time learning performance. How to choose the baseline function is currently an open question. I propose that the baseline function should be chosen to minimize the mean squared error (MSE) of the gradient estimates. Under slightly idealistic conditions, I prove that this choice gives the tightest bound on the suboptimality of the algorithm obtainable from the standard analysis of stochastic gradient ascent. Unfortunately, the MSE-minimizing baseline function depends on the transition probability kernel

$\mathbb{T}$ of the MDP, so the agent can not directly compute it. The final contribution of this project is to show that the MSE-minimizing baseline can be estimated from only observable quantities.

4.1 Policy Gradient Methods

Policy gradient methods are learning algorithms for MDPs based on stochastic gradient ascent. We will consider two specific algorithms, one for learning to maximize the total reward in episodic MDPs, and another for learning to maximize the long-term average reward in ergodic MDPs.

To apply stochastic gradient ascent, we need to express the problem of choosing good policies in terms of maximizing a function $f : \mathbb{R}^d \to \mathbb{R}$. One way to accomplish this is to choose a scheme for representing policies as vectors in $\mathbb{R}^d$. Such a scheme is called a policy parameterization, and is a function $\pi : \mathbb{R}^d \to \Pi$ that gives us a policy for each *parameter vector* $\theta \in \mathbb{R}^d$. The composition of a policy parameterization $\pi : \mathbb{R}^d \to \Pi$ and a formal learning objective $J : \Pi \to \mathbb{R}$ gives us a map $\theta \mapsto J(\pi(\theta))$ which is a suitable objective function for stochastic gradient ascent. To simplify notation, I will write $J(\theta)$ to mean $J(\pi(\theta))$. The set of policies that can be represented by a parameterization π is given by $\pi(\mathbb{R}^d) = \{\pi(\theta) : \theta \in \mathbb{R}^d\}$. Maximizing $J(\theta)$ over $\mathbb{R}^d$ is equivalent to maximizing $J(\pi)$ over the set of policies $\pi(\mathbb{R}^d)$.

Typically, not every Markov policy will be representable by a policy parameterization (i.e., $\pi(\mathbb{R}^d)$ is a strict subset of Π). This is actually an advantage of policy gradient methods. Intuitively, the difficulty of finding good parameters for a parameterization scales with the number of parameters. For many real-world problems, the reinforcement learning practitioner can design a policy parameterization with only a few parameters but which can still represent nearly optimal policies. This allows the practitioner to leverage their understanding of a problem to get better performance in practice.

Policy gradient methods were enabled by the development of techniques for generating random vectors that are equal in expectation to the gradient of

the learning objective. I will loosely call such a random vector an estimated gradient. The remainder of this section presents two gradient estimation techniques: one for the total reward in episodic MDPs originally due to Williams [W1992] and one for the average reward in ergodic MDPs due to Sutton *et al.* [SMSM2000]. The presentation is slightly modified from the original sources to better fit with this thesis.

Both gradient estimates have the same basic structure. To estimate the gradient $\nabla J(\theta)$, the agent follows policy $\pi(\theta)$ for a short period of time. For any state-action pair (x, a), the policy gradient $\nabla_\theta \pi(x, a, \theta)$ is the direction in parameter-space that the agent would move the parameter vector θ to increase the probability of choosing action a from state x. The learning objective gradient estimate is the sum of the policy gradients over the state-action pairs visited during the trial period, each scaled by a term that depends on how well the agent performed following that action (the details depend on J) divided by the probability of choosing action a from state x. Intuitively, the effect of adding this gradient estimate to the agent's parameter vector is to increase the probability of the actions that performed well, and to increase even more the probability of actions that performed well and are rarely chosen. The following two theorems give detailed constructions of the gradient estimates and show that they are equal in expectation to the gradient of the learning objective.

Theorem 4.1. *Let $\pi : \mathbb{R}^d \to \Pi$ be a parametric policy for an episodic MDP and let $\theta \in \mathbb{R}^d$ be any parameter vector. Let $(X_t^\theta, A_t^\theta, R_t^\theta)_{t=1}^\infty$ be the sequence of states, actions, and rewards obtained by following policy $\pi(\theta)$ for a single episode, let T^θ be the first time the terminal state is entered, and let $G_t^\theta = \sum_{s=t}^{T^\theta} R_s^\theta$ be the total reward earned after time t. Finally, let $\varphi_t^\theta = \frac{\nabla_\theta \pi(X_t^\theta, A_t^\theta, \theta)}{\pi(X_t^\theta, A_t^\theta, \theta)}$ be the so-called vector of compatible features at time t. Then, the random vector*

$$\nabla_\theta = \sum_{t=1}^{T^\theta - 1} \varphi_t^\theta G_t^\theta$$

satisfies $\mathbb{E}[\nabla_\theta] = \nabla J_{\text{total}}(\theta)$.

Since the policy parameterization π is chosen by the reinforcement learn-

ing practitioner, the policy gradient $\nabla_\theta \pi(x, a, \theta)$ can be computed by the agent. All other quantities that appear in the gradient estimate from Theorem 4.1 are observable by the agent. Further, since the gradient estimate is a function of the current parameter vector θ, whenever θ is itself random, we have the property $\mathbb{E}[\nabla_\theta \,|\, \theta] = \nabla J_{\text{total}}(\theta)$. Therefore, we can use these gradient estimates in a simple policy gradient method that updates the policy parameters once after each episode. Pseudocode is given in Algorithm 4.

Input: step-size $\eta > 0$.
Choose $\theta_1 \in \mathbb{R}^d$ arbitrarily;
for *each episode index* $\tau = 1, 2, \ldots$ **do**
    Run one episode following $\pi(\theta)$ until the terminal state is reached;
    Compute ∇_{θ_τ} according to Theorem 4.1;
    Set $\theta_{\tau+1} = \theta_\tau + \eta \nabla_{\theta_\tau}$;
end

Algorithm 4: Policy Gradient Method for Episodic MDPs

Theorem 4.2. *Let $\pi : \mathbb{R}^d \to \Pi$ be a parametric policy for an ergodic MDP and let $\theta \in \mathbb{R}^d$ be any parameter vector. Let X^θ be randomly sampled from the stationary distribution $\nu(\pi(\theta))$, A^θ be sampled from $\pi(x, \cdot, \theta)$, and $\varphi^\theta = \frac{\nabla_\theta \pi(X^\theta, A^\theta, \theta)}{\pi(X^\theta, A^\theta, \theta)}$ be the so-called vector of compatible features. Then the random vector*

$$\nabla_\theta = \varphi^\theta Q_{\text{avg}}(X^\theta, A^\theta, \theta)$$

satisfies $\mathbb{E}[\nabla_\theta] = \nabla J_{\text{avg}}(\theta)$.

Unlike in the episodic case, this gradient estimate depends on quantities that are unknown to the agent. First, the action value function depends on the MDP transition probability kernel $\mathbb{T}$, which is unknown. In place of the true action value function, we can use an estimate (for example, the estimate from linear Sarsa(λ). For a modern account of Sarsa(λ), see Section 7.5 of [SB1998]). Using estimated action values introduces a small bias into the gradient estimates, but its effect on the performance of stochastic gradient ascent is small. Second, the random variable X^θ in the statement

of the theorem is drawn from the stationary distribution $\nu(\pi(\theta))$ which the agent doesn't know and can't directly sample from. Rather than drawing a sample from $\nu(\pi(\theta))$, the agent can simply use the state they visit while interacting with the MDP. In practice, the distribution over states at time t is close enough to the stationary distribution that this also introduces only a small bias. So, in the average reward setting, the agent is able to compute a random vector which is nearly an unbiased estimate of the gradient of $J_{\text{avg}}(\theta)$. Pseudocode for an policy gradient method based on this nearly unbiased gradient estimate is given in Algorithm 5.

Input: step-size $\eta > 0$.
Choose $\theta_1 \in \mathbb{R}^d$ arbitrarily;
Initialize action value estimate $\hat{q}$;
for *each time* $t = 1, 2, \ldots$ **do**
    Receive state X_t from the environment;
    Sample action A_t from $\pi(X_t, \cdot, \theta)$;
    Receive reward R_t;
    Compute ∇_{θ_t} according to Theorem 4.2 using X_t, A_t, and the estimated action value function $\hat{q}$;
    Set $\theta_{t+1} = \theta_t + \eta \nabla_{\theta_t}$;
    Update the estimated action value function estimate $\hat{q}$;
end

Algorithm 5: Policy Gradient Method for Ergodic MDPs

4.2 Baseline Functions

In both of the gradient estimation techniques from the previous section, the value zero plays a special role. In the total reward setting (and the average reward setting is similar), each time t in an episode contributes an update to the parameter vector that increase the probability of choosing the action A_t from state X_t. The scale of this update is proportional to the difference $G_t = G_t - 0$, where G_t is the total reward following action

A_t. One consequence is that if G_t is positive, then the update will make the action more probable, and if it is negative, then the update will make the action less probable. This behaviour is strange, since there is nothing special about earning zero total reward. In fact, some MDPs have only negative rewards and, in this case, the agent never directly increases the probability of choosing good actions. They are only increased as a side-effect of decreasing the probability of worse actions more aggressively. This raises the following questions: Can we compare the total reward G_t to a value other than zero? And how does the choice affect the performance of the policy gradient method?

We will see below that, rather than comparing to zero, we can compare to any baseline value that depends on the state X_t and the agent's parameter vector θ. The function $b : \mathcal{X} \to \mathbb{R}$ that maps each state to the baseline value in that state is called the baseline function. The following two results show how to incorporate the baseline function into the gradient estimates from the previous section.

Corollary 4.3. *Let $\pi : \mathbb{R}^d \to \Pi$ be a policy parameterization for an episodic MDP, $\theta \in \mathbb{R}^d$ be any parameter vector, and $b : \mathcal{X} \to \mathbb{R}$ be any baseline. Assume that $\nabla_\theta \pi(x_{\text{term}}, a, \theta) = 0$ for all actions a and parameter vectors θ (since the actions chosen in the terminal state have no effects, the policy parameterization can always be made to satisfy this condition). Further, suppose that $\mathbb{E}[T^\theta] < \infty$ for all parameter vectors θ. Using the notation of Theorem 4.1, the random vector*

$$\nabla_\theta^b = \sum_{t=1}^{T^\theta - 1} \varphi_t^\theta (G_t^\theta - b(X_t^\theta))$$

satisfies $\mathbb{E}[\nabla_\theta^b] = \nabla J_{\text{total}}(\theta)$.

Proof. From Theorem 4.1, it suffices to show that

$$\mathbb{E}\left[\sum_{t=1}^{T^\theta - 1} \varphi_t^\theta b(X_t^\theta)\right] = 0.$$

For any time $t \geq T^\theta$, we have that $X_t^\theta = x_{\text{term}}$ and therefore $\varphi_t^\theta = 0$. From this, it follows that

$$\sum_{t=1}^{T^\theta - 1} \varphi_t^\theta b(X_t^\theta) = \sum_{t=1}^{\infty} \varphi_t^\theta b(X_t^\theta).$$

Consider any component θ_i of θ. Letting $\varphi_{t,i}^\theta$ denote the i^{th} component of φ_t^θ, we have

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{\infty} \left|\varphi_{t,i}^\theta b(X_t^\theta)\right|\right] &= \mathbb{E}\left[\sum_{t=1}^{\infty} \left| \frac{\frac{\partial}{\partial \theta_i}\pi(X_t^\theta, A_t^\theta, \theta)}{\pi(X_t^\theta, A_t^\theta, \theta)} b(X_t^\theta) \right|\right] \\
&= \mathbb{E}\left[\sum_{x,a} \sum_{t=1}^{\infty} \mathbb{I}\left\{X_t^\theta = x, A_t^\theta = a\right\} \left| \frac{\frac{\partial}{\partial \theta_i}\pi(x, a, \theta)}{\pi(x, a, \theta)} b(x) \right|\right] \\
&= \mathbb{E}\left[\sum_{\substack{x \neq x_{\text{term}}, \\ a}} \sum_{t=1}^{\infty} \mathbb{I}\left\{X_t^\theta = x, A_t^\theta = a\right\} \left| \frac{\frac{\partial}{\partial \theta_i}\pi(x, a, \theta)}{\pi(x, a, \theta)} b(x) \right|\right] \\
&= \sum_{\substack{x \neq x_{\text{term}}, \\ a}} \left(\left| \frac{\frac{\partial}{\partial \theta_i}\pi(x, a, \theta)}{\pi(x, a, \theta)} b(x) \right| \cdot \mathbb{E}\left[\sum_{t=1}^{\infty} \mathbb{I}\left\{X_t^\theta = x, A_t^\theta = a\right\}\right] \right) \\
&\leq \sum_{\substack{x \neq x_{\text{term}}, \\ a}} \left(\left| \frac{\frac{\partial}{\partial \theta_i}\pi(x, a, \theta)}{\pi(x, a, \theta)} b(x) \right| \cdot \mathbb{E}[T^\theta] \right) \\
&< \infty,
\end{aligned}$$

where in line 3 we used the fact that $\frac{\partial}{\partial \theta_i}\pi(x_{\text{term}}, a, \theta) = 0$. Therefore, by Fubini's theorem, we have for each i

$$\mathbb{E}\left[\sum_{t=1}^{\infty} \varphi_{t,i}^\theta b(X_t^\theta)\right] = \sum_{t=1}^{\infty} \mathbb{E}[\varphi_{t,i}^\theta b(X_t^\theta)].$$

In vector form, this gives

$$\mathbb{E}\left[\sum_{t=1}^{\infty} \varphi_t^\theta b(X_t^\theta)\right] = \sum_{t=1}^{\infty} \mathbb{E}[\varphi_t^\theta b(X_t^\theta)].$$

Using the shorthand $p_t(x,a) = \mathbb{P}(X_t = x, A_t = a_t)$, $p_t(a \,|\, x) = \mathbb{P}(A_t = a \,|\, X_t = x)$, and $p_t(x) = \mathbb{P}(X_t = x)$, we can rewrite the expectation as a sum over the possible state-action pairs

$$\begin{aligned}
\mathbb{E}\big[\varphi_t^\theta b(X_t^\theta)\big] &= \sum_{x,a} p_t(x,a) \frac{\nabla_\theta \pi(x,a,\theta)}{\pi(x,a,\theta)} b(x) \\
&= \sum_{x,a} p_t(a \,|\, x) p_t(x) \frac{\nabla_\theta \pi(x,a,\theta)}{\pi(x,a,\theta)} b(x) \\
&= \sum_{x,a} \pi(x,a,\theta) p_t(x) \frac{\nabla_\theta \pi(x,a,\theta)}{\pi(x,a,\theta)} b(x) \\
&= \sum_{x} p_t(x) b(x) \sum_{a} \nabla_\theta \pi(x,a,\theta) \\
&= 0,
\end{aligned}$$

where in the last line we used that $\sum_a \nabla_\theta \pi(x,a,\theta) = \nabla_\theta 1 = 0$. We have shown that

$$\mathbb{E}\bigg[\sum_{t=1}^{T^\theta - 1} \varphi_t^\theta b(X_t^\theta)\bigg] = 0,$$

as required. □

Corollary 4.4. *Let $\pi : \mathbb{R}^d \to \Pi$ be a policy parameterization for an ergodic MDP, $\theta \in \mathbb{R}^d$ be any parameter vector, and $b : \mathcal{X} \to \mathbb{R}$ be any baseline function. Using the notation of Theorem 4.2, the random vector*

$$\nabla_\theta^b = \varphi^\theta \big(Q_{\text{avg}}(X^\theta, A^\theta, \theta) - b(X^\theta)\big)$$

satisfies $\mathbb{E}\nabla_\theta^b = \nabla J_{\text{avg}}(\theta)$.

Proof. Again it suffices to show that $\mathbb{E}[\varphi^\theta b(X^\theta)] = 0$. Using the shorthand $p(x,a) = \mathbb{P}(X^\theta = x, A^\theta = a)$, $p(a \,|\, x) = \mathbb{P}(A^\theta = a \,|\, X^\theta = x)$ and $p(x) =$

$\mathbb{P}(X^\theta = x)$, we can rewrite the expectation as a sum

$$\begin{aligned}\mathbb{E}\big[\varphi^\theta b(X^\theta)\big] &= \sum_{x,a} p(x,a)\frac{\nabla_\theta \pi(x,a,\theta)}{\pi(x,a,\theta)} b(x) \\ &= \sum_x p(x) b(x) \sum_a \nabla_\theta \pi(x,a,\theta) \\ &= 0.\end{aligned}$$

□

Notice that in the proof of Corollary 4.4, the expectation of $\varphi^\theta b(X^\theta)$ is zero independently of the distribution over X^θ. Therefore, even when we compute the gradient estimate with the state visited by the agent, which is not distributed according to the stationary distribution, the baseline function introduces no additional bias.

Two commonly used baseline functions are the constantly zero baseline and the value function of the current policy: $b(x) = V(x,\theta)$, where $V(x,\theta) = V_{\text{total}}(x, \pi(\theta))$ in the total reward setting and $V(x,\theta) = V_{\text{avg}}(x,\pi(\theta))$ in the average reward setting. The value function baseline seems like a natural choice, since it compares the agent's sampled performance to her expected performance. If she performs better than expected, then she should increase the probability of the actions tried and decrease them otherwise.

A hint that these two baseline functions are sub-optimal is that they do not depend in any way on the policy parameterization. That is, given two policy parameterizations $\pi_1 : \mathbb{R}^{d_1} \to \Pi$ and $\pi_2 : \mathbb{R}^{d_2} \to \Pi$, there may be parameter vectors $\theta_1 \in \mathbb{R}^{d_1}$ and $\theta_2 \in \mathbb{R}^{d_2}$ such that $\pi_1(\theta_1) = \pi_2(\theta_2)$. In this case, the value function and constantly zero baseline will have the same value for both policy parameterizations in every state. But, since the baseline function's purpose is to modify the parameter updates, it would be surprising if we should choose the baseline in a way that does not depend on the particular parameterization used.

[GBB2004] have proposed that the baseline function should be chosen to minimize the variance of the gradient estimates (specifically, the trace of the covariance matrix). They consider learning in partially observable

MDPs and a different performance gradient estimate than the two described in the previous section, so their results do not directly carry over to the two cases studied in this project, though there are strong similarities. They derive closed-form expressions for the variance minimizing baseline, a theory that analyzes the variance of different baseline functions, and algorithms for estimating the variance minimizing baseline. To the best of my knowledge, they do not show that the variance minimizing baseline is a good or optimal choice.

The goal in the rest of the chapter is to explore the connection between the baseline function and the performance of policy gradient methods, and to decide which baseline function gives the best performance.

4.3 MSE Minimizing Baseline

In this section I argue that the baseline function should be chosen to minimize the mean squared error of the performance gradient estimates. This section derives a closed form expression for the MSE minimizing baseline, which reveals an interesting connection to the value function baseline. I also show that when the learning objective is a concave function of the policy parameters, the MSE-minimizing baseline gives the best possible bound on the agent's total expected sub-optimality obtainable from a standard analysis of stochastic gradient ascent.

First, I show that choosing the baseline to minimize the MSE of the gradient estimates is equivalent to choosing it to minimize the trace of the covariance matrix of the estimates, which is equivalent to choosing it to minimize the second moment of the gradient estimates. This equivalence is useful because minimizing the MSE makes inuitive sense, minimizing the second moment is the easiest to work with formally, and minimizing the trace of the covariance matrix shows that this idea is equivalent to minimizing the variance, which was already proposed by Greensmith, Bartlett, and Baxter. The idea of minimizing the variance is not new, but the following analysis and estimation techniques are.

Lemma 4.5. *Let $\mu \in \mathbb{R}^d$ be any vector and suppose for each function $b : \mathcal{X} \to \mathbb{R}$, we have a random vector ∇^b such that $\mathbb{E}[\nabla^b] = \mu$ for all b. Then*

$$\operatorname*{argmin}_{b:\mathcal{X}\to\mathbb{R}} \operatorname{tr} \operatorname{Cov}(\nabla^b) = \operatorname*{argmin}_{b:\mathcal{X}\to\mathbb{R}} \mathbb{E}\big[\big\|\nabla^b - \mu\big\|_2^2\big] = \operatorname*{argmin}_{b:\mathcal{X}\to\mathbb{R}} \mathbb{E}\big[\big\|\nabla^b\big\|_2^2\big].$$

Proof. This result is a consequence of the fact that $\mathbb{E}[\nabla^b]$ does not depend on b. Using the trace rotation equality (that is, $\operatorname{tr}(AB) = \operatorname{tr}(BA)$) and the definition of the covariance matrix, we have

$$\begin{aligned}
\operatorname{tr} \operatorname{Cov}(\nabla^b) &= \operatorname{tr}(\mathbb{E}[(\nabla^b - \mu)(\nabla^b - \mu)^\top]) \\
&= \mathbb{E}[\operatorname{tr}\big((\nabla^b - \mu)(\nabla^b - \mu)^\top\big)] \\
&= \mathbb{E}[(\nabla^b - \mu)^\top(\nabla^b - \mu)] \\
&= \mathbb{E}\big[\big\|\nabla^b - \mu\big\|_2^2\big]
\end{aligned}$$

which proves the first equivalence. Expanding the definition of the squared 2-norm, we have

$$\begin{aligned}
\mathbb{E}\big[\big\|\nabla^b - \mu\big\|_2^2\big] &= \mathbb{E}\big[\big\|\nabla_b\big\|_2^2\big] + \big\|\mu\big\|_2^2 - 2\mathbb{E}[\nabla_b^\top]\mu \\
&= \mathbb{E}\big[\big\|\nabla_b\big\|_2^2\big] - \big\|\mu\big\|_2^2.
\end{aligned}$$

It follows that $\mathbb{E}\big[\big\|\nabla^b - \mu\big\|_2^2\big]$ and $\mathbb{E}\big[\big\|\nabla_b\big\|_2^2\big]$ differ by a constant that does not depend on b. Therefore, the same function b will minimize both expressions. □

Applying Lemma 4.5 to the case where $\mu = \nabla J(\theta)$ and ∇^b is a gradient estimate with baseline b shows that minimizing the MSE is equivalent to minimizing the trace of the covariance matrix, which is equivalent to minimizing the second moment.

4.3.1 Regret Bound from the MSE Minimizing Baseline

In this section, I prove that the MSE-minimizing baseline has good theoretical properties. Suppose that the learning objective J is a concave function of the policy parameters and we use a policy gradient method to produce

a sequence of parameter vectors $\theta_1, \ldots, \theta_T$. Let θ^* be any parameter vector that maximizes J. We can use Theorem 3.3 to upper bound the sum $\sum_{t=1}^{T} \mathbb{E}[J(\theta) - J(\theta_t)]$, which is one measure of the agent's learning performance until time T. The upper bound that we get is

$$\sum_{t=1}^{T} \mathbb{E}[J(\theta) - J(\theta_t)] \leq \frac{B^2}{\eta} + \eta T G^2,$$

where η is the step size used in the policy gradient method, B is an upper bound on $\|\theta^*\|_2$, and G^2 is an upper bound on the second moments of the stochastic gradient estimates. Setting the step size according to $\eta = B/(G\sqrt{T})$ gives the best bound of $2BG\sqrt{T}$. The gradient estimates only appear in this bound through their second moments, and minimizing the second moments gives the best bound. Since minimizing the second moments of the gradient estimates is equivalent to minimizing the MSE, the MSE-minimizing baseline gives the best possible regret bound from Theorem 3.3.

The requirement that J be a concave function of the policy parameters is almost never satisfied. However, it is often the case that J will be concave in a neighbourhood of its local maxima. In this case, once the algorithm enters one of these neighbourhoods, the above analysis holds if we replace θ^* with the local maxima.

4.3.2 MSE Minimizing Baseline for Average Reward

This section derives a closed form expression for the MSE minimizing baseline for the average reward learning objective. The derivation for the average reward is given before the derivation for the total reward because it is considerably simpler and uses the same ideas.

Theorem 4.6. *Let $\pi : \mathbb{R}^d \to \Pi$ be a policy parameterization for an ergodic MDP and let $\theta \in \mathbb{R}^d$ be any parameter vector. Then the function*

$$b(x) = \sum_a w(x, a, \theta) Q_{\text{avg}}(x, a, \theta),$$

where

$$w(x,a,\theta) = \frac{\tilde{w}(x,a,\theta)}{\sum_{a'} \tilde{w}(x,a',\theta)} \quad \textit{and} \quad \tilde{w}(x,a,\theta) = \frac{\left\|\nabla_\theta \pi(x,a,\theta)\right\|_2^2}{\pi(x,a,\theta)},$$

is the baseline function that minimizes the MSE of the gradient estimate ∇_θ^b *in Corollary 4.4.*

Proof. By Lemma 4.5, we can equivalently find the minimizer of the second moment of the gradient estimate. That is, we want to solve the optimization problem

$$\operatorname*{argmin}_{b:\mathcal{X}\to\mathbb{R}} \mathbb{E}\left[\left\|\varphi^\theta(Q_{\text{avg}}(X^\theta, A^\theta, \theta) - b(X^\theta))\right\|_2^2\right].$$

The general approach is as follows: by writing the expectation as a sum, we see that it is separable over the values $b(x)$, so we are free to optimize each $b(x)$ independently. Further, the contribution of each $b(x)$ to the second moment of the gradient estimate of ∇_θ^b is quadratic in $b(x)$ and therefore the minimizer can easily be computed.

First, we write the second moment as a sum and show it separates over the values $b(x)$. Let $p(x,a)$, $p(a \mid x)$, and $p(x)$ be shorthand for the probabilities $\mathbb{P}(X^\theta = x, A^\theta = a)$, $\mathbb{P}(A^\theta = a \mid X^\theta = x)$, and $\mathbb{P}(X^\theta = x)$, respectively. Then

$$\begin{aligned}
&\mathbb{E}\left[\left\|\varphi^\theta(Q_{\text{avg}}(X^\theta, A^\theta, \theta) - b(X^\theta))\right\|_2^2\right] \\
&\qquad = \mathbb{E}\left[\left\|\varphi^\theta\right\|_2^2 (Q_{\text{avg}}(X^\theta, A^\theta, \theta) - b(X^\theta))^2\right] \\
&\qquad = \sum_{x,a} p(x,a) \frac{\left\|\nabla_\theta \pi(x,a,\theta)\right\|_2^2}{\pi(x,a,\theta)^2} (Q_{\text{avg}}(x,a,\theta) - b(x))^2 \\
&\qquad = \sum_{x,a} p(a \mid x) p(x) \frac{\tilde{w}(x,a,\theta)}{\pi(x,a,\theta)} (Q_{\text{avg}}(x,a,\theta) - b(x))^2 \\
&\qquad = \sum_x p(x) \sum_a \tilde{w}(x,a,\theta) (Q_{\text{avg}}(x,a,\theta) - b(x))^2.
\end{aligned}$$

For each state x, the value $b(x)$ only appears in exactly one term of the sum over x. Since there are no constraints between the $b(x)$, we are free to minimize each term independently. Therefore, we can express the MSE-

minimizing baseline point-wise as

$$\begin{aligned} b(x) &= \operatorname*{argmin}_{c\in\mathbb{R}} p(x) \sum_a \tilde{w}(x,a,\theta)\big(Q_{\text{avg}}(x,a,\theta) - c\big)^2 \\ &= \sum_a w(x,a,\theta) Q_{\text{avg}}(x,a,\theta), \end{aligned}$$

which completes the proof. □

There is an interesting connection between the MSE-minimizing baseline and the value function: both are weighted averages of the action values. Since $\tilde{w}(x,a,\theta)$ is non-negative for each action a, the weights $w(x,a,\theta)$ form a probability distribution over the actions and therefore the MSE-minimizing baseline $b(x)$ is a weighted average of the action values. Similarly, the Bellman equation shows that the value function is a weighted average of the action values:

$$V_{\text{avg}}(x,\theta) = \sum_a \pi(x,a,\theta) Q_{\text{avg}}(x,a,\theta).$$

The only difference between these two baseline functions is the weighting used in the average.

It is also interesting to notice that the value function baseline minimizes the second moment of the quantity $Q_{\text{avg}}(X^\theta, A^\theta, \theta) - b(X^\theta)$. The MSE-minimizing baseline uses the modified weights $w(x,a,\theta)$ in place of the action selection probabilities in order to instead minimize the second moment of the gradient estimate $\varphi^\theta(Q_{\text{avg}}(X^\theta, A^\theta, \theta) - b(X^\theta))$.

4.3.3 MSE Minimizing Baseline for Total Reward

This section derives expressions for two different baseline functions for the total reward learning objective. The first is the baseline function that truly minimizes the MSE, but it has a complicated form due to the correlations between the states and actions visited during a single episode. The second baseline is derived by ignoring the correlations between states, has a much simpler form, and may still come close to minimizing the MSE in practice. I will first present the exact MSE minimizing baseline, followed by the

approximation.

Theorem 4.7. *Let $\pi : \mathbb{R}^d \to \Pi$ be a policy parameterization for an episodic MDP and let $\theta \in \mathbb{R}^d$ be any parameter vector. Then the function*

$$b(x) = \frac{\sum_{t=1}^{\infty} \mathbb{E}\big[\mathbb{I}\left\{X_t^\theta = x\right\} \nabla_\theta^\top \varphi_t^\theta\big]}{\sum_{t=1}^{\infty} \mathbb{E}\big[\mathbb{I}\left\{X_t^\theta = x\right\} \left\|\varphi_t^\theta\right\|_2^2\big]},$$

where $\nabla_\theta = \nabla_\theta^0$ is the baseline function that minimizes the MSE of the gradient estimate ∇_θ^b in Corollary 4.3.

Proof. Let $(X_t^\theta, A_t^\theta, R_t^\theta)_{t=1}^\infty$ be the random sequence of states, actions, and rewards obtained by following $\pi(\theta)$, let T^θ be the first time the terminal state is entered, $G_t^\theta = \sum_{s=t}^{T^\theta - 1} R_s^\theta$ be the total reward earned after time t, and $\varphi_t^\theta = \frac{\nabla_\theta \pi(X_t^\theta, A_t^\theta, \theta)}{\pi(X_t^\theta, A_t^\theta, \theta)}$ be the vector of compatible features at time t. We can rewrite the gradient estimate from Corollary 4.3 as

$$\begin{aligned}
\nabla_\theta^b &= \sum_{t=1}^{\infty} \varphi_t^\theta (G_t^\theta - b(X_t^\theta)) \\
&= \sum_{t=1}^{\infty} \varphi_t^\theta G_t - \sum_{t=1}^{\infty} \varphi_t^\theta b(X_t^\theta) \\
&= \nabla_\theta - \sum_{t=1}^{\infty} \varphi_t^\theta b(X_t^\theta),
\end{aligned}$$

where ∇_θ is the gradient estimate with the constantly zero baseline function.

We can therefore decompose the second moment as follows

$$\begin{aligned}
\mathbb{E}\big[\big\|\nabla_\theta^b\big\|_2^2\big] &= \mathbb{E}\left[\left(\nabla_\theta - \sum_{t=1}^{\infty} \varphi_t^\theta b(X_t^\theta)\right)^\top \left(\nabla_\theta - \sum_{t=1}^{\infty} \varphi_t^\theta b(X_t^\theta)\right)\right] \\
&= \mathbb{E}\big[\big\|\nabla_\theta\big\|_2^2\big] - 2\sum_{t=1}^{\infty} \mathbb{E}\big[\nabla_\theta^\top \varphi_t^\theta b(X_t^\theta)\big] + \sum_{t=1}^{\infty}\sum_{s=1}^{\infty} \mathbb{E}\big[{\varphi_t^\theta}^\top \varphi_s^\theta b(X_t^\theta) b(X_s^\theta)\big].
\end{aligned} \tag{4.1}$$

The double sum in (4.1) can be simplified using the following observation. Let s and t be any two times with $s < t$. Since Markov policies choose

actions independently of the history of states and actions, the probability of choosing action a from state X_t^θ at time t does not depend on the state and action visited at time s. Formally, for any states x and x', and any actions a and a', we have

$$\mathbb{P}(A_t^\theta = a \,|\, X_t^\theta = x, X_s^\theta = x', A_s^\theta = a') = \mathbb{P}(A_t^\theta = a | X_t^\theta = x) = \pi(x, a, \theta).$$

Therefore, we can factor the joint probability of taking action a' from state x' at time s and later taking action a from state x at time t as follows:

$$\begin{aligned}
&\mathbb{P}(X_t^\theta = x, A_t^\theta = a, X_s^\theta = x', A_s^\theta = a') \\
&\quad = \mathbb{P}(A_t^\theta = a \,|\, X_t^\theta = x, X_s^\theta = x', A_s^\theta = a')\mathbb{P}(X_t^\theta = x \,|\, X_s^\theta = x', A_s^\theta = a') \\
&\qquad \cdot \mathbb{P}(A_s^\theta = a' \,|\, X_s^\theta = x')\mathbb{P}(X_s^\theta = x') \\
&\quad = \pi(x, a, \theta)\pi(x', a', \theta)\mathbb{P}(X_s^\theta = x')\mathbb{P}(X_t^\theta = x \,|\, X_s^\theta = x', A_s^\theta = a').
\end{aligned}$$

Note that this trick does not work when the time s follows time t, because then knowing the state X_s^θ gives you information about what action was taken at time t. Using this factorization, for any times $s < t$, we have

$$\begin{aligned}
&\mathbb{E}[{\varphi_t^\theta}^\top \varphi_s^\theta b(X_t^\theta) b(X_s^\theta)] \\
&\quad = \sum_{x,a,x',a'} \Bigg(\mathbb{P}(X_t^\theta = x, A_t^\theta = a, X_s^\theta = x', A_s^\theta = a') \\
&\qquad\qquad \cdot \frac{\nabla_\theta \pi(x, a, \theta)^\top}{\pi(x, a, \theta)} \frac{\nabla_\theta \pi(x', a', \theta)}{\pi(x', a', \theta)} b(x) b(x') \Bigg) \\
&\quad = \sum_{x,x',a'} \Bigg(\mathbb{P}(X_s^\theta = x')\mathbb{P}(X_t^\theta = x \,|\, X_s^\theta = x', A_s^\theta = a') \\
&\qquad\qquad \cdot \Big(\sum_a \nabla_\theta \pi(x, a, \theta)^\top \Big) \nabla_\theta \pi(x', a', \theta) b(x) b(x') \Bigg) \\
&\quad = 0,
\end{aligned}$$

where the last line uses the fact that $\sum_a \nabla_\theta \pi(x, a, \theta) = \nabla_\theta 1 = 0$. The expression is symmetric in the times t and s, so an identical argument can be used to show that $\mathbb{E}[{\varphi_t^\theta}^\top \varphi_x^\theta b(X_t^\theta) b(X_s^\theta)] = 0$ when $s > t$. Therefore, the

only non-zero terms of the double sum from (4.1) are the terms when $s = t$, and therefore we can write the second moment as

$$\begin{aligned}\mathbb{E}\big[\big\|\nabla_\theta^b\big\|_2^2\big] &= \mathbb{E}\big[\big\|\nabla_\theta\big\|_2^2\big] - 2\sum_{t=1}^{\infty}\mathbb{E}\big[\nabla_\theta^\top \varphi_t^\theta b(X_t^\theta)\big] + \sum_{t=1}^{\infty}\mathbb{E}\big[\big\|\varphi_t^\theta\big\|_2^2 b(X_t^\theta)^2\big] \\ &= \mathbb{E}\big[\big\|\nabla_\theta\big\|_2^2\big] + \sum_x \sum_{t=1}^{\infty}\Bigg(\mathbb{E}\big[\mathbb{I}\left\{X_t^\theta = x\right\}\big\|\varphi_t^\theta\big\|_2^2\big] b(x)^2 \\ &\qquad\qquad - 2\mathbb{E}\big[\mathbb{I}\left\{X_t^\theta = x\right\}\nabla_\theta^\top \varphi_t^\theta\big] b(x)\Bigg).\end{aligned}$$

The last line above is obtained by first summing over the states, and then summing over the times in which state x was visited, and shows that the second moment is again separable over the states x.

Since the second moment is separable over the values $b(x)$, we are free to minimize each independently. Again, the contribution of each $b(x)$ to the second moment is an easily-minimized quadratic expression in $b(x)$. The MSE-minimizing baseline is therefore given by

$$b(x) = \frac{\sum_{t=1}^{\infty}\mathbb{E}\big[\mathbb{I}\left\{X_t^\theta = x\right\}\nabla_\theta^\top \varphi_t^\theta\big]}{\sum_{t=1}^{\infty}\mathbb{E}\big[\mathbb{I}\left\{X_t^\theta = x\right\}\big\|\varphi_t^\theta\big\|_2^2\big]}.$$

□

The above expression for the MSE minimizing baseline may not be very useful, since it may not be easy to compute even if we knew the MDP transition probability kernel $\mathbb{T}$. Both the form and derivation of this baseline function are complicated because of the correlations between the states and actions visited at different times during a single episode.

We can derive a much simpler alternative baseline that may still go a long way towards minimizing the MSE. The idea is, rather than minimizing the second moment of the complete gradient estimate $\nabla_\theta^b = \sum_{t=1}^{T^\theta}\varphi_t^\theta(G_t^\theta - b(X_t^\theta))$, we can try to minimize the second moment of each term in the sum independently. The following result shows that a much simpler baseline function minimizes the second moment of $\varphi_t^\theta(G_t^\theta - b(X_t^\theta))$ for all times t.

Theorem 4.8. *Let $\pi : \mathbb{R}^d \to \Pi$ be a policy parameterization for an episodic MDP and let $\theta \in \mathbb{R}^d$ be any parameter vector. Then the function*

$$b(x) = \sum_a w(x, a, \theta) Q_{\text{total}}(x, a, \theta)$$

where

$$w(x, a, \theta) = \frac{\tilde{w}(x, a, \theta)}{\sum_{a'} \tilde{w}(x, a', \theta)} \qquad \text{and} \qquad \tilde{w}(x, a, \theta) = \frac{\|\nabla_\theta \pi(x, a, \theta)\|_2^2}{\pi(x, a, \theta)}$$

simultaneously minimizes the second moment of $\varphi_t^\theta(G_t^\theta - b(X_t^\theta))$ for all times t.

Proof. Fix any time t and let X_t^θ and A_t^θ be the state and action at time t when following policy $\pi(\theta)$. Let $\varphi_t^\theta = \frac{\nabla_\theta \pi(X_t^\theta, A_t^\theta, \theta)}{\pi(X_t^\theta, A_t^\theta, \theta)}$ be the vector of compatible features at time t and G_t^θ be the total reward following action A_t^θ. Our goal is to find the baseline function $b : \mathcal{X} \to \mathbb{R}$ that minimizes

$$\mathbb{E}\big[\big\|\varphi_t^\theta(G_t^\theta - b(X_t^\theta))\big\|_2^2\big].$$

Again, the general strategy is to express the objective as a separable sum over the states x and to solve for each $b(x)$ independently. Let $p_t(x, a) = \mathbb{P}(X_t = x, A_t = a)$, $p_t(a \,|\, x) = \mathbb{P}(A_t = a \,|\, X_t = x)$, and $p_t(x) = \mathbb{P}(X_t = x)$. Using the fact that $\mathbb{E}[G_t^\theta \,|\, X_t^\theta = x, A_t^\theta = a] = Q_{\text{total}}(x, a, \theta)$ and the definition of $\tilde{w}$ from the statement of the theorem, we have

$$\begin{aligned}
&\mathbb{E}\big[\big\|\varphi_t^\theta(G_t^\theta - b(X_t^\theta))\big\|_2^2\big] \\
&\quad = \mathbb{E}\big[\big\|\varphi_t^\theta\big\|_2^2 (G_t^\theta - b(X_t^\theta)^2\big] \\
&\quad = \sum_{x,a} p_t(x, a) \frac{\big\|\nabla_\theta \pi(x, a, \theta)\big\|_2^2}{\pi(x, a, \theta)^2} (Q_{\text{total}}(x, a, \theta) - b(x))^2 \\
&\quad = \sum_{x,a} p_t(a \,|\, x) p_t(x) \frac{\tilde{w}(x, a, \theta)}{\pi(x, a, \theta)} (Q_{\text{total}}(x, a, \theta) - b(x))^2 \\
&\quad = \sum_x p_t(x) \sum_a \tilde{w}(x, a, \theta) (Q_{\text{total}}(x, a, \theta) - b(x))^2.
\end{aligned}$$

From this we see that the second moment is separable over the states and again the contribution of each state is quadratic in $b(x)$. Therefore, we can choose each $b(x)$ independently as follows

$$\begin{aligned} b(x) &= \operatorname*{argmin}_{c \in \mathbb{R}} p(x) \sum_a \tilde{w}(x, a, \theta) Q_{\text{total}}(x, a, \theta) - b(x))^2 \\ &= \sum_a w(x, a, \theta) Q_{\text{total}}(x, a, \theta). \end{aligned}$$

Since the above baseline does not depend on the time index t, it simultaneously minimizes the second moment of each $\varphi_t^\theta(G_t^\theta - b(X_t^\theta))$. □

This approximate MSE-minimizing baseline shows that, modulo the correlations between times in an episode, the MSE minimizing baseline in the total reward setting has the same form as in the average reward setting.

4.4 Estimating the MSE Minimizing Baseline

It may be easier to directly estimate the MSE minimizing baseline than to estimate the unknown quantities in the closed-form expressions from the previous section. For example, the action-value function (which appears in two of the three baselines) is a function from state-action pairs to real numbers, while the baseline function is only a map from states to real numbers. Since the baseline function has a simpler form, it may be possible to estimate it more easily and from less data than the action value function. This section describes a stochastic-gradient method for directly estimating the MSE minimizing baselines from experience.

We will represent our baseline estimates as linear function approximators. Specifically, we will fix a map $\psi : \mathcal{X} \to \mathbb{R}^n$ which maps each state to a *feature vector*. Our baseline estimate will be a linear function of the feature vector: $b(x, w) = \psi(x)^\top w$, where $w \in \mathbb{R}^n$ is the baseline parameter vector. Our goal is to find the parameter vector $w \in \mathbb{R}^d$ that minimizes the MSE of the gradient estimate $\nabla_\theta^w = \nabla_\theta^{b(\cdot,w)}$, which is equivalent to minimizing the second moment.

For both the average and total reward settings, we will show that the second moment is a convex quadratic function of the weights w used in the baseline function. Minimizing the second moment is equivalent to maximizing the negative second moment, which is a concave function. Therefore, if we can construct unbiased random estimates of the gradient $\nabla_w \mathbb{E}\big[\big\|\nabla_\theta^w\big\|_2^2\big]$, then we can use stochastic gradient ascent to directly approximate the MSE minimizing baselines. The following theorems show that it is possible to compute unbiased random estimates of the gradient by interacting with the environment.

Theorem 4.9. *Let $\pi : \mathbb{R}^d \to \Pi$ be a policy parameterization for an ergodic MDP, $\theta \in \mathbb{R}^d$ be any policy parameter vector, $\psi : \mathcal{X} \to \mathbb{R}^n$ be a baseline feature map, and $w \in \mathbb{R}^d$ be any baseline parameter vector. Then the map $w \mapsto \mathbb{E}\big[\big\|\nabla_\theta^w\big\|\big]$ is a convex quadratic function of w and the random vector*

$$D_\theta^w = 2\big\|\varphi^\theta\big\|_2^2 \psi^\theta \psi^{\theta\top} w - 2\psi^\theta \varphi^{\theta\top} \nabla_\theta^0$$

satisfies $\mathbb{E}[D_\theta^w] = \nabla_w \mathbb{E}[\big\|\nabla_\theta^w\big\|_2^2]$, where $\psi^\theta = \psi(X^\theta)$ and ∇_θ^w is the gradient estimate from Corollary 4.4 with baseline $b(x) = \psi(x)^\top w$.

Proof. We can rewrite ∇_θ^w as follows

$$\begin{aligned}\nabla_\theta^w &= \varphi^\theta (Q_{\text{avg}}(X^\theta, A^\theta, \theta) - \psi^{\theta\top} w) \\ &= \nabla_\theta^0 - \varphi^\theta \psi^{\theta\top} w.\end{aligned}$$

With this, we have

$$\begin{aligned}\mathbb{E}\big[\big\|\nabla_\theta^w\big\|_2^2\big] &= \mathbb{E}\big[(\nabla_\theta^0 - \varphi^\theta \psi^{\theta\top} w)^\top (\nabla_\theta^0 - \varphi^\theta \psi^{\theta\top} w)\big] \\ &= w^\top \mathbb{E}[\big\|\varphi^\theta\big\|_2^2 \psi^\theta \psi^{\theta\top}] w - 2\mathbb{E}[\nabla_\theta^0 (\varphi^\theta \psi^{\theta\top})] w + \mathbb{E}[\big\|\nabla_\theta^0\big\|].\end{aligned}$$

This is a quadratic equation in w. Since the second moment is bounded

below by 0, it follows that it must also be convex. With this, we have

$$\begin{aligned}\nabla_w \mathbb{E}\big[\big\|\nabla_\theta^w\big\|_2^2\big] &= \nabla_w \Big\{ w^\top \mathbb{E}[\big\|\varphi^\theta\big\|_2^2 \psi^\theta \psi^{\theta\top}] w - 2\mathbb{E}[\nabla_\theta^0 (\varphi^\theta \psi^{\theta\top})] w + \mathbb{E}[\big\|\nabla_\theta^0\big\|] \Big\} \\ &= \mathbb{E}\left[2\big\|\varphi^\theta\big\|_2^2 \psi^\theta \psi^{\theta\top} - 2\psi^\theta {\varphi^\theta}^\top \nabla_\theta^0 \right]\end{aligned}$$

and it follows that

$$D_\theta^w = 2\big\|\varphi^\theta\big\|_2^2 \psi^\theta \psi^{\theta\top} - 2\psi^\theta {\varphi^\theta}^\top \nabla_\theta^0$$

is an unbiased estimate of the gradient $\nabla_w \mathbb{E}\big[\big\|\nabla_\theta^w\big\|_2^2\big]$. □

All quantities in D_θ^w are observable by the agent, so the agent can produce samples from D_θ^w. Pseudocode for a policy gradient method that uses the above estimate of the baseline function is given in Algorithm 6

Input: policy step-size $\eta > 0$, baseline step-size $\alpha > 0$
Choose $\theta_1 \in \mathbb{R}^d$ arbitrarily;
Choose $w_1 \in \mathbb{R}^n$ arbitrarily;
Initialize action value estimate $\hat{q}$;
for *each time* $t = 1, 2, \ldots$ **do**
    Receive state X_t from the environment;
    Sample action A_t from $\pi(X_t, \cdot, \theta)$;
    Receive reward R_t;
    Compute $\nabla_{\theta_t}^{w_t}$ according to Corollary 4.4 using X_t, A_t, the estimated action value function $\hat{q}$, and the baseline function $b(x) = \psi(x)^\top w_t$;
    Set $\theta_{t+1} = \theta_t + \eta \nabla_{\theta_t}$;
    Compute $D_{\theta_t}^{w_t}$ according to Theorem 4.9;
    Set $w_{t+1} = w_t - \alpha D_{\theta_t}^{w_t}$;
    Update the estimated action value function estimate $\hat{q}$;
end

Algorithm 6: Policy gradient method for ergodic MDPs with a linear approximation to the MSE minimizing baseline.

Theorem 4.10. *Let $\pi : \mathbb{R}^d \to \Pi$ be a policy parameterization for an episodic MDP, $\theta \in \mathbb{R}^d$ be any policy parameter vector, $\psi : \mathcal{X} \to \mathbb{R}^n$ be a baseline feature map, and $w \in \mathbb{R}^n$ be any baseline parameter vector. Assume that $\nabla_\theta \pi(x_{\text{term}}, a, \theta) = 0$ for all actions a and parameter vectors θ (the parameterization can always be chosen so that this property is satisfied). Then the map $w \mapsto \mathbb{E}\big[\big\|\nabla_\theta^w\big\|\big]$ is a convex quadratic function of w and the random vector*

$$D_\theta^w = 2 \sum_{t=1}^{T^\theta - 1} \psi_t^\theta \big(\big\|\varphi_t^\theta\big\|_2^2 \psi_t^{\theta\top} w - \varphi_t^{\theta\top} \nabla_\theta^0 \big)$$

satisfies $\mathbb{E}[D_\theta^w] = \nabla_w \mathbb{E}\big[\big\|\nabla_\theta^w\big\|_2^2\big]$, where $\psi_t^\theta = \psi(X_t^\theta)$ and ∇_θ^w is the gradient estimate from Corollary 4.3 with baseline $b(x) = \psi(x)^\top w$.

Proof. We can rewrite ∇_θ^w as follows

$$\begin{aligned}
\nabla_\theta^w &= \sum_{t=1}^{T^\theta - 1} \varphi_t^\theta (G_t^\theta - \psi_t^{\theta\top} w) \\
&= \sum_{t=1}^{\infty} \varphi_t^\theta G_t - \Big(\sum_{t=1}^{\infty} \varphi_t^\theta \psi_t^{\theta\top} \Big) w \\
&= \nabla_\theta^0 - M^\theta w,
\end{aligned}$$

where $M^\theta = \sum_{t=1}^\infty \varphi_t^\theta \psi_t^{\theta\top}$. We can use this to write the second moment as

$$\begin{aligned}
\mathbb{E}\big[\big\|\nabla_\theta^w\big\|_2^2\big] &= \mathbb{E}\big[(\nabla_\theta^0 - M^\theta w)^\top (\nabla_\theta^0 - M^\theta w)\big] \\
&= w^\top \mathbb{E}\big[M^{\theta\top} M^\theta\big] w - 2\mathbb{E}\big[\nabla_\theta^{0\top} M^\theta\big] w + \mathbb{E}\big[\big\|\nabla_\theta^0\big\|_2^2\big].
\end{aligned}$$

Again this is a quadratic form in w and, since the second moment is bounded below, it must be convex.

Before taking the gradient, we can simplify the matrix of coefficients $\mathbb{E}\big[M^{\theta\top} M^\theta\big]$ in a way very similar to the simplification of the double sum in Theorem 4.7. Recall that for any times $s < t$, we have the following

$$\begin{aligned}
&\mathbb{P}(X_t^\theta = x, A_t^\theta = a, X_s^\theta = x', A_s^\theta = a') \\
&\qquad = \pi(x, a, \theta) \pi(x', a', \theta) \mathbb{P}(X_t^\theta = x \,|\, X_s^\theta = x', A_s^\theta = a') \mathbb{P}(X_s^\theta = x').
\end{aligned}$$

Therefore, whenever $s < t$, we have

$$\begin{aligned}
\mathbb{E}[\psi_t^\theta \varphi_t^{\theta\top} \varphi_s^\theta \psi_s^{\theta\top}] &= \sum_{x,a,x',a'} \mathbb{P}(X_t^\theta = x, A_t^\theta = a, X_s^\theta = x', A_s^\theta = a') \\
&\qquad \cdot \psi(x) \frac{\nabla_\theta \pi(x,a,\theta)^\top}{\pi(x,a,\theta)} \frac{\nabla_\theta \pi(x',a',\theta)}{\pi(x',a',\theta)} \psi(x')^\top \\
&= \sum_{x,x',a'} \mathbb{P}(X_s^\theta = x') \mathbb{P}(X_t^\theta = x \,|\, X_s^\theta = x', A_s\theta = a') \\
&\qquad \cdot \psi(x) \big(\sum_a \nabla_\theta \pi(x,a,\theta)^\top \big) \nabla_\theta \pi(x',a',\theta) \psi(x')^\top \\
&= 0.
\end{aligned}$$

Similarly, when $s > 0$, we have $\mathbb{E}[\psi_t^\theta \varphi_t^{\theta\top} \varphi_s^\theta \psi_s^{\theta\top}] = 0$. Therefore,

$$\begin{aligned}
\mathbb{E}\big[M^{\theta\top} M^\theta\big] &= \sum_{t=1}^\infty \sum_{s=1}^\infty \mathbb{E}\big[\psi_t^\theta \varphi_t^{\theta\top} \varphi_s^\theta \psi_s^{\theta\top}\big] \\
&= \sum_{t=1}^\infty \mathbb{E}\big[\big\|\varphi_t^\theta\big\|_2^2 \psi_t^\theta \psi_t^{\theta\top}\big].
\end{aligned}$$

Finally, since $\nabla_\theta \pi(x_{\text{term}}, a, \theta) = 0$, we have that $\varphi_t^\theta = 0$ whenever $X_t^\theta = x_{\text{term}}$, therefore

$$\mathbb{E}\big[M^{\theta\top} M^\theta\big] = \mathbb{E}\bigg[\sum_{t=1}^{T^\theta} \big\|\varphi_t^\theta\big\|_2^2 \psi_t^\theta \psi_t^{\theta\top}\bigg]$$

and

$$\mathbb{E}\big[M^{\theta\top} \nabla_\theta^0\big] = \mathbb{E}\bigg[\bigg(\sum_{t=1}^{T^\theta} \varphi_t^\theta \psi_t^{\theta\top}\bigg) \nabla_\theta^0\bigg]$$

Substituting these equalities into the expression for the second moment, we have

$$\mathbb{E}\big[\big\|\nabla_\theta^w\big\|_2^2\big] = w^\top \mathbb{E}\bigg[\sum_{t=1}^{T^\theta} \big\|\varphi_t^\theta\big\|_2^2 \psi_t^\theta \psi_t^{\theta\top}\bigg] w - 2\bigg[\bigg(\sum_{t=1}^{T^\theta} \varphi_t^\theta \psi_t^{\theta\top}\bigg) \nabla_\theta^0\bigg]^\top w + \mathbb{E}\big[\big\|\nabla_\theta^0\big\|_2^2\big].$$

Taking the gradient of this expression and exchanging the gradient and

expectation gives that

$$D_\theta^w = 2\sum_{t=1}^{T^\theta} \psi_t^\theta \left(\left\| \varphi_t^\theta \right\|_2^2 {\psi_t^\theta}^\top w - {\varphi_t^\theta}^\top \nabla_\theta^0 \right)$$

satisfies $\mathbb{E}[D_\theta^w] = \nabla_w \mathbb{E}\left[\left\| \nabla_\theta^w \right\|_2^2 \right]$. □

It is worth noting that the gradient estimate from Theorem 4.10 can be computed in linear time in the length of the episode. The gradient estimate ∇_θ^0 can be computed in the first pass, and D_θ^w can be computed in a second pass.

Pseudocode for a policy gradient method that uses the above estimate of the baseline function is given in Algorithm 7

Input: policy step-size $\eta > 0$, baseline step-size $\alpha > 0$
Choose $\theta_1 \in \mathbb{R}^d$ arbitrarily;
Choose $w_1 \in \mathbb{R}^n$ arbitrarily;
for *each episode index* $\tau = 1, 2, \ldots$ **do**
    Run one episode following $\pi(\theta)$ until the terminal state is reached;
    Compute $\nabla_{\theta_\tau}^{w_\tau}$ according to Corollary 4.3 using the baseline $b(x) = \psi(x)^\top w_\tau$;
    Set $\theta_{\tau+1} = \theta_\tau + \eta \nabla_{\theta_\tau}$;
    Compute $D_{\theta_\tau}^{w_\tau}$ according to Theorem 4.10;
    Set $w_{\tau+1} = w_\tau - \alpha D_{\theta_\tau}^{w_\tau}$.
end

Algorithm 7: Policy gradient method for episodic MDPs with a linear approximation to the MSE minimizing baseline.

4.5 Experiments

This section presents an empirical comparison of the three baseline functions discussed in this thesis: the always zero baseline, the value function, and the MSE minimizing baseline. The goal of these experiments is to answer

the question: Does the baseline function have a significant impact on performance? And if so, does one of the baseline functions consistently give better performance than the others?

Earlier in this chapter I showed that when the formal learning objective is a concave function of the policy parameters, then we can upper bound the agent's expected regret. This regret bound only depended on the baseline through the second moments of the performance gradient estimates. Therefore, the best regret bound is obtained by choosing the baseline to minimize the second moment, which we saw was equivalent to minimizing the MSE. We also saw that the best regret bound after T updates is obtained by setting the step size to be

$$\eta(T) = \frac{B}{G\sqrt{T}},$$

where G^2 is an upper bound on the second moments of the first T gradient estimates, and B is an upper bound on the 2-norm of the unknown optimal parameter vector. Therefore, in the concave setting, the step size which gives the best regret bound scales inversely proportional to G. My hypothesis is that even when the learning objective is not a concave function of the policy parameters, choosing the baseline to minimize the MSE (and therefore the second moment) is a good choice. This hypothesis would be supported by the experiments if the MSE baseline gives the highest performance and if its maximum performance is achieved with a higher step size than for the other baselines. All but one of the following experiments support this hypothesis.

Policy gradient methods have two significant parameters: the step size and the baseline function. The expected performance depends on the pair of parameter settings. To study the effect of the baseline function, I estimate the expected performance when using each baseline with a wide range of step sizes. For each baseline, this results in a graph showing the performance of the given baseline as a function of the step size. This kind of parameter study involves running the algorithm many more times than would be done in practice. In practice, the parameters might be chosen according to a rule of thumb or based on some brief experiment and then the algorithm may be run only once with those settings. In these experiments, however,

we run the algorithm for many parameter settings and, for each parameter setting, we do enough runs to accurately estimate the expected total reward. While these parameter studies do not necessarily resemble reinforcement learning in practice, they allow us to confidently compare the baselines and to understand how their performance depends on the step size. A strong result would be to find that one baseline outperforms the rest for every step size. In this case, no matter how you choose the step size, you should always use the winning baseline. A weaker result would be to find that the maximum performance of one baseline (maximizing over the step sizes) is higher than the maximum performances of the other baselines. This result is weaker, since it might not be easy to find good step sizes in practice.

The experiments compare the three baseline functions in two different test-beds. I borrowed the first test-bed, called the ten-armed test-bed, from Rich Sutton and Andy Barto's book [SB1998]. In the ten-armed test-bed, the MDP has a single state and ten actions, each having rewards drawn from a Gaussian distribution with unit standard deviation. Such an MDP is called a bandit problem, in reference to multi-armed bandits or slot machines. An episode in a bandit problem consists of choosing a single arm, and the agent's goal is to maximize her total reward over a fixed number of episodes. In all the following experiments, the number of episodes is taken to be 20. Rather than fixing a single bandit for the comparison, we can randomly produce many distinct bandits by sampling the mean payout for each arm from a standard Gaussian distribution. We compare the average performance over a large number of randomly generated bandits to reduce the chance that the results are an artifact of one specific bandit.

MDPs with a single state are missing many properties of typical MDPs. For example, the action value function in a single-state MDP does not depend on the agent's policy at all. I designed the second test bed, called the triangle test-bed, to be similar to the ten-armed test-bed but with more than one state. Again, the payout for each state-action pair will be a Gaussian with unit standard deviation and mean sampled from a standard Gaussian distribution. The states are organized in a triangle with $R = 5$ rows, where there are r states in the r^{th} row. The starting state is the unique state in

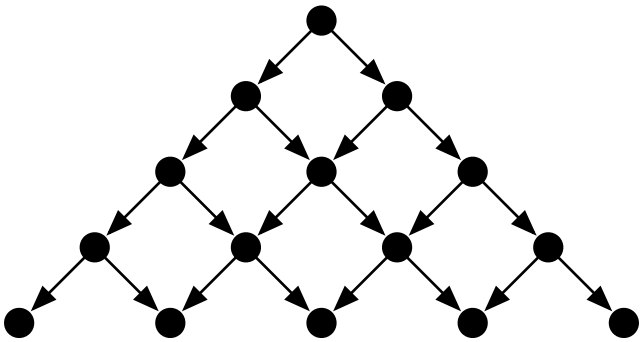

Figure 4.1: Each black circle represents a state in a triangular MDP and each arrow represents the deterministic transition resulting from the LEFT action or RIGHT action, depending on whether the arrow points left or right. Starting from the top state, the agent chooses to move either LEFT or RIGHT until she reaches the bottom row.

the first row and there are two actions, LEFT and RIGHT, which each move the agent from its current state to the state below and left, or below and right, respectively. Figure 4.1 shows the layout of the states and the available actions. As in the ten-armed test-bed, the agent is allowed to perform a fixed number of episodes on each randomly generated triangle MDP, and her goal is to maximize total reward. In all of the following experiments, the agent is given 50 episodes to interact with each instance of the triangle MDP.

I use a similar policy parameterization in both test-beds. The policy parameterization is a mixture of the uniform policy, which chooses each action with equal probability, and the so-called Gibbs policy, which is a common policy parameterization when there are a small number of states and actions. In the Gibbs policy, the agent stores a weight, or preference, for each state-action pair. The weights are stored in a $|\mathcal{X} \times \mathcal{A}|$-dimensional vector θ, and we write $\theta(x, a)$ to denote the weight associated to action a in state x. The Gibbs policy is parameterized by the agent's preferences in

the following way:

$$\pi_{\text{Gibbs}}(x, a, \theta) = \frac{\exp(\theta(x, a))}{\sum_{a'} \exp(\theta(x, a'))},$$

where the sum is taken over all actions. Under the Gibbs policy, the agent prefers to choose actions with a high weight and the action selection probabilities are invariant to adding a constant to all weights. For a mixing constant ϵ, which in all experiments I take to be 0.05, the policy parameterization that I use is given by

$$\pi(x, a, \theta) = \epsilon/|\mathcal{A}| + (1 - \epsilon)\pi_{\text{Gibbs}}(x, a, \theta).$$

The mixture constant ϵ is not a parameter of the policy that is controlled by the agent. The reason for using a mixture of the Gibbs policy and the uniform policy is that, in the mixture, the minimum action selection probability is $\epsilon/|\mathcal{A}| > 0$. Having a parameter-independent lower bound on the action selection probabilities ensures that the agent will continue to explore the available actions for every possible parameter vector. This helps to avoid situations where the agent sets the probability of choosing an action to zero early in an episode based on an unlucky reward. It is a straight forward calculation to check that the gradient of the Gibbs policy with respect to the weight vector θ is given by

$$\nabla_\theta \pi_{\text{Gibbs}}(x, a, \theta) = \pi_{\text{Gibbs}}(x, a, \theta)(e(x, a) - \pi_{\text{Gibbs}}(x, \theta)),$$

where $e(x, a)$ is the $(x, a)^{\text{th}}$ standard-basis vector and $\pi_{\text{Gibbs}}(x, \theta)$ is the vector whose $(x', a')^{\text{th}}$ entry is equal to $\mathbb{I}\{x = x'\}\, \pi_{\text{Gibbs}}(x', a', \theta)$. From this, the gradient of the mixed policy is given by

$$\begin{aligned}\nabla_\theta \pi(x, a, \theta) &= (1 - \epsilon)\nabla_\theta \pi_{\text{Gibbs}}(x, a, \theta) \\ &= (1 - \epsilon)\pi_{\text{Gibbs}}(x, a, \theta)(e(x, a) - \pi_{\text{Gibbs}}(x, \theta)).\end{aligned}$$

Since we have closed form expressions for the action selection probabilities and their gradient with respect to the policy parameters, it is easy to

compute the stochastic gradient estimates discussed in this chapter. In all experiments, the weight vector θ is always initialized to the zero vector, which results in the uniform policy.

One practical complication is that the value function and the MSE minimizing baseline are both unknown to the computer agent, since they depend on the unknown MDP transition probability kernel. An agent solving real-world problems must estimate these functions in order to use them as baselines. The experimental challenge is that, if we compare estimated baselines, we can't be sure that a difference in performance is due to the choice of baseline and not the quality of the estimation. Even though unknowable baseline functions are not usable in practice, measuring their performance is still useful because it motivates the search for good approximation methods for the best theoretical baselines. For this reason, the experiments compare not only the estimated baselines, but also their true values. In both of the test-beds, we have perfect knowledge of the transition probability kernel (though we do not share this information with the agent). Using this knowledge, we can give the computer agent access to an oracle that produces the true MSE minimizing baseline function, or the true value function.

Comparison of True Baseline Functions: First, I present the results of the comparison in the setting where the agent has access to the true MSE minimizing baseline and the true value function. This is the simplest setting, since there are no parameters related to baseline estimation that need to be tuned. Figure 4.2 shows the estimated performance in both the ten-armed and triangle test-beds. In both test-beds, the expected total reward for each parameter setting is estimated by taking the sample mean of 1,000,000 independent runs. These parameter studies give statistically significant support to my hypothesis, since the MSE minimizing baseline gives better performance than the zero baseline and the value function across a wide range of step sizes, and it attains its highest performance at a higher step-size than the other baselines.

It appears, however, that using either the value function or the MSE minimizing baseline gives only a small improvement over using the always-

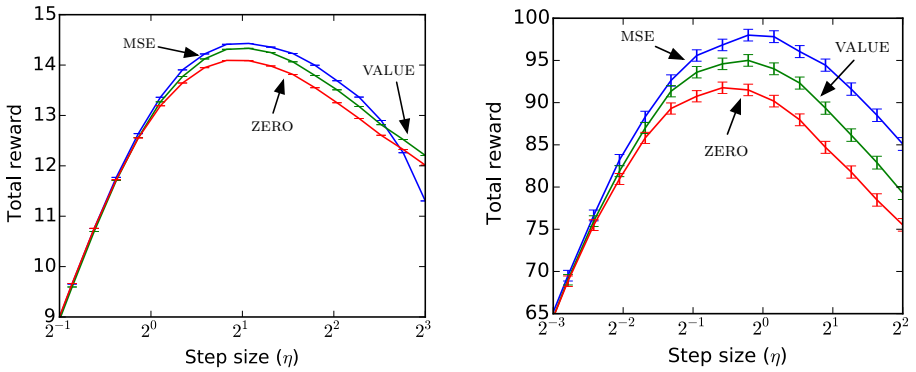

Figure 4.2: Comparison of the total reward earned by the episodic policy gradient method when using the true baseline functions. The error bars show 3 standard errors in the mean. The left figure shows performance in the ten-armed test-bed. The right figure shows performance in the triangle test-bed.

zero baseline, which is equivalent to using no baseline at all. For the ten-armed test-bed, the average reward for each action is drawn from a standard Gaussian random variable and therefore zero is a good guess at the average payout of the arms. We might expect that if the mean payouts were shifted up or down, the performance of the zero baseline may deteriorate, since zero is no longer a good estimate of the mean arm payout. In contrast, the updates made by both the value function and MSE minimizing baselines do not change when a constant is added to all rewards. The situation is similar in the triangle test-bed, since the expected payout of a path through the states is also zero. Figure 4.3 shows the performance of the baselines in both test-beds when the rewards are shifted up or down by 10. In this case, we see a significant improvement when using non-zero baselines. It is difficult to see the difference between the value function and the MSE minimizing baseline, but since these methods are invariant to translations in the rewards, their difference is exactly the same as in the mean-zero case.

The above comparisons show that for the two test-beds, it is important

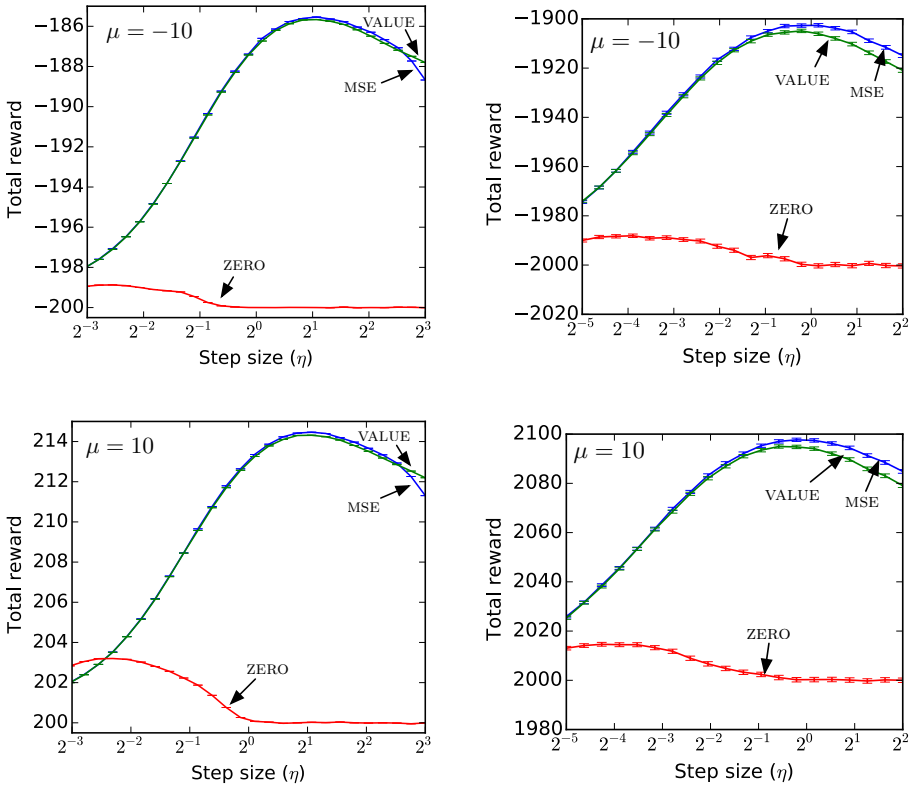

Figure 4.3: Comparisons of the three baselines in the ten-armed test-bed and the triangle test-bed. The mean reward for each action is drawn from a Gaussian distribution with mean μ and unit standard deviation. In the first row, $\mu = -10$ and in the second row $\mu = 10$. The left column shows estimates of the expected total reward in the ten-armed test-bed and the second column shows the same for the triangle test-bed. Again, the error bars show 3 standard errors in the mean.

to use a non-zero baseline whenever the mean payout per episode is far from zero. Moreover, the MSE minimizing baseline consistently gave better performance than the other baselines for a wide range of step sizes, and its maximum performance was obtained at a higher step size than for the other baselines, which supports my hypothesis.

Comparison of Estimated Baseline Functions Next, I will present the results of the comparison of the estimated baseline functions. This case is slightly more complicated than when using the true baseline functions, since we need to choose any parameters that the estimation techniques have. Rather than carefully optimizing the parameters for the estimation techniques, I tried to choose the parameters in a way that would be realistic in practice. First, I will describe the estimation techniques and the parameter choices, followed by a comparison of the different baselines. The names I use for each baseline estimate are prefixed with either the letter Q or G, which indicates how they are estimated as will be described shortly.

I proposed two different techniques for estimating the MSE minimizing baseline. In the first estimation, we ignored the correlations between different time-steps in each episode, which gave rise to an approximate form of the MSE minimizing baseline that is a weighted average of the action value function. When the MDP has only a single state, there are no correlations to ignore and this approximation is exact. Given an estimate of the action value function, which can be obtained in various standard ways, we can substitute the estimated action values into the closed form approximation of the MSE minimizing baseline. This estimate is appealing because its only parameters are those of the action value estimation technique, which in many cases can be chosen according to rules-of-thumb. I will refer to this estimate as the QMSE baseline (Q for action-values). The second estimation was obtained by performing stochastic gradient descent to estimate the MSE minimizing baseline directly from the observed sequences of states, actions, and rewards. This estimation is appealing because it does not ignore the correlation between time steps in the episode, but one draw back is that its step size parameter is difficult to tune. I will refer to this estimate as the

GMSE (G for stochastic gradient descent).

I expect that the choice of the step size for the stochastic gradient descent algorithm used to compute the GMSE baseline, denoted by η_{bl} will have a similar effect on the performance of the agent for all policy gradient step sizes η. Therefore, to set η_{bl} for each test bed, I ran the agent with policy gradient step size $\eta = 0.9$ for the correct number of episodes (20 in the ten-armed test-bed and 50 in the triangle test-bed) 1000 times and chose the baseline step-size from the set $\{0.001, 0.01, 0.1, 1.0\}$ that maximized performance. The best parameter settings were $\eta_{\text{bl}} = 0.01$ in the ten-armed test bed and $\eta_{\text{bl}} = 0.1$ in the triangle test-bed.

There are standard techniques for estimating the value function of a policy in an MDP. Rather than estimating the value function directly, though, I use the fact that the value function is a weighted average of the action value function. This gives more accurate estimates in the ten-armed test-bed, since the action values do not depend on the agent's current policy. I will refer to this as the QVALUE baseline.

To estimate the action value function in the ten-armed test-bed, I use the fact that the action value function does not depend on the agent's policy, since there is only one state. In this case, a good estimate of the action value for a given action is to take the sample average of the observed rewards for that action. For actions that have not yet been tried, a default value of 0 is used. The only parameter of this estimation technique for the action values is the default value. Since it only influences performance early in learning, I did not tune the default value.

In the triangle test-bed, I use the Sarsa(λ) algorithm to estimate the action value functions that are passed into the two action-value oriented baseline estimates. Sarsa(λ) has two parameters, a step size α and an eligibility trace parameter λ. Again, I expect that the Sarsa(λ) parameters should affect the agent's performance similarly for all policy-gradient step sizes and all baselines. For this reason, I chose the parameters α and λ by running the policy gradient method with fixed step size $\eta = 0.9$ for 50 episodes 1000 times, and chose the parameters that gave the smallest average squared error in the action-value estimates at the end of the 50 episodes.

Of all pairs of α in $\{0.001, 0.01, 0.1, 1.0\}$ and λ in $\{0.001, 0.01, 0.1, 1.0\}$ the best setting was to take $\alpha = \lambda = 0.1$.

Figure 4.4 shows the parameter studies for each of the estimated baselines in the two test-beds. As before, I also present the results when the rewards are shifted up or down by 10. The results of this experiment tell a different story than what we saw for the true baseline functions. Let μ denote the amount that the rewards are shifted by. In the first experiments where we compared the true baseline functions, the MSE minimizing baseline gave the best performance across a wide range of parameters. In this setting, however, the baseline with the best performance depends on the step size and which baseline achieves the highest performance for an optimized step size depends on the value of μ. These results do not support my hypothesis and were surprising because I expected the relative performance of the non-zero baselines to be the same independent of the shift μ.

The differences in performance between the non-zero baselines for the various values of μ can be explained by an interesting property of policy gradient methods. Consider the bandit problem and suppose that our baseline is substantially larger than the rewards. Suppose the agent chooses a random action A and receives reward R. Then the term $(R - b)$, where b is the baseline value, is negative with high probability the agent will reduce the probability of choosing action A, even if it was the best action. Since the action selection probabilities must sum to one, the probability of the other actions will be increased. On the following episode, the agent will be more likely to choose an action other than A, even if A was the best available action. In this way, having a baseline that underestimates the rewards encourages systematic exploration of the actions. On the other hand, if the baseline is substantially lower than the rewards, the probability of choosing action A will always be increased, even if it was the worst action. On the following episodes, the agent will be more likely to choose the same action again. This asymmetry between having baselines that are too high or too low suggests that it is better to have an underestimate, which results in exploration, rather than an overestimate, which results in less exploration and more erratic updates to the parameter vector.

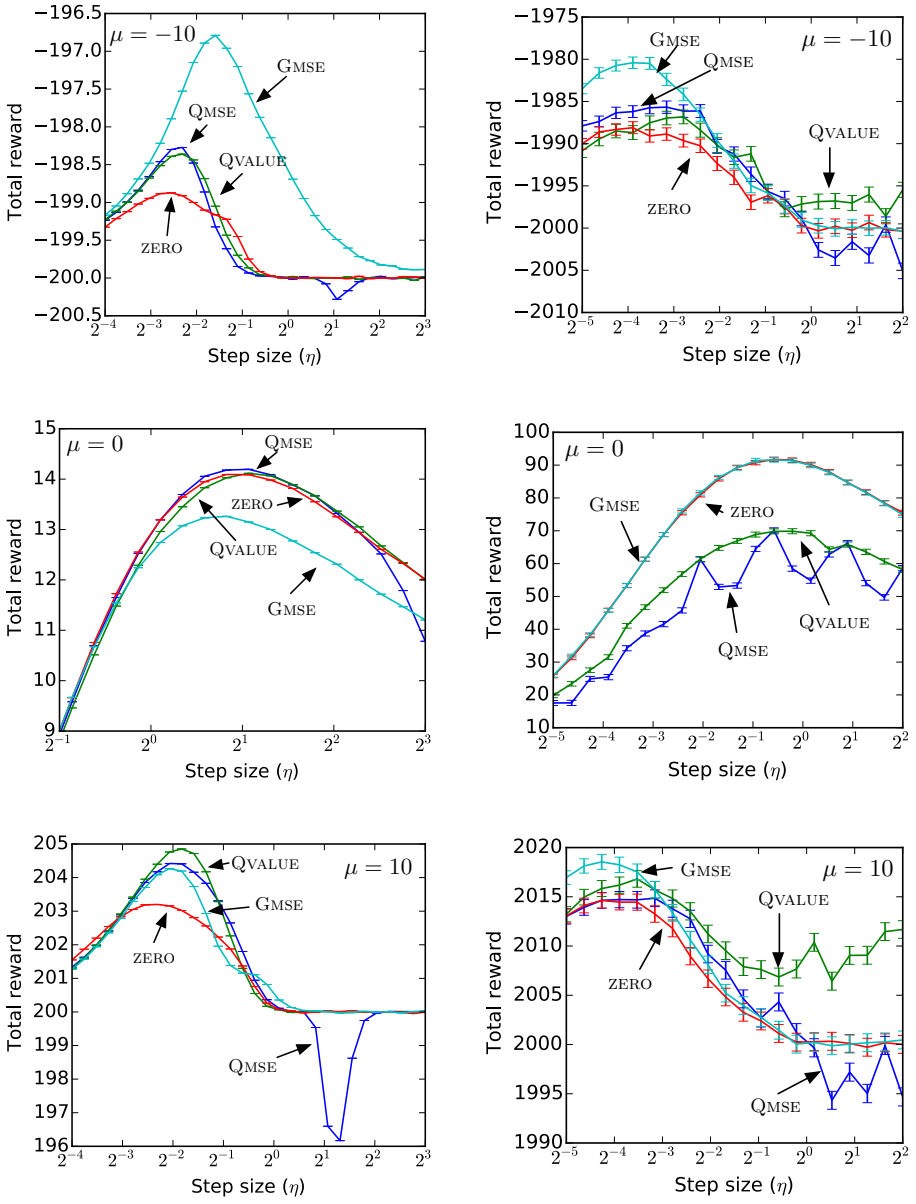

Figure 4.4: Comparisons of the four estimated baselines in the ten-armed the triangle test-beds. The mean reward for each action is drawn from a Gaussian distribution with mean μ and unit standard deviation. In the first row, $\mu = 10$ in the second row, $\mu = 0$, and in the third row $\mu = -10$. The left column shows estimates of the expected total reward in the ten-armed test-bed and the second column shows the same for the triangle test-bed. Again, the error bars show 3 standard errors in the mean.

But why should this asymmetry change which of the non-zero baselines performs best? The reason is that both of the non-zero baselines are weighted averages of the action values. In the case of the QVALUE baseline, the weights are exactly the action selection probabilities, so it places high weight on the actions that will have the most reliable action value estimates. On the other hand, the weights in the QMSE baseline are proportional to $\|\nabla_\theta \pi(x, a, \theta)\|_2^2 / \pi(x, a, \theta)$. Since the denominator scales inversely with the action selection probabilities, the weighted average depends more heavily on the actions that are infrequently selected. Therefore, when the initial action value estimates are very high, as is the case when $\mu = -10$, we expect there to be enough exploration for both estimated baselines to become accurate. In this case, the MSE minimizing baselines performs better. But when $\mu = 10$, the amount of exploration is reduced and therefore the value function estimate becomes more accurate than for the MSE baselines. This is one possible explanation for the difference between the $\mu = 10$ and $\mu = -10$ cases.

To test this hypothesis I ran the experiments again, but this time I initialized the starting action value estimate to be a better estimate than 0. In the ten-armed test-bed, I pull each arm once and use the sampled reward as the initial estimate of the action value, instead of using the default value of zero. For the triangle test-bed, I compute the true value function and initialize the action value estimate with this instead. In principle, I could have run several episodes using Sarsa(λ) to compute a more realistic initial estimate of the action value function for the triangle MDP, but using the true values requires less computation and has essentially the same result. Further, even though this initialization is slightly unrealistic, it shouldn't favour any baseline function. Results for this experiment are shown in Figure 4.5. These results are very similar to those for the true baseline functions and support my hypothesis. The lesson from this experiment is that when using policy gradient methods, we should be careful to initialize the baseline function in a reasonable way so that the agent's policy does not become too focused early on, independently of the observed rewards.

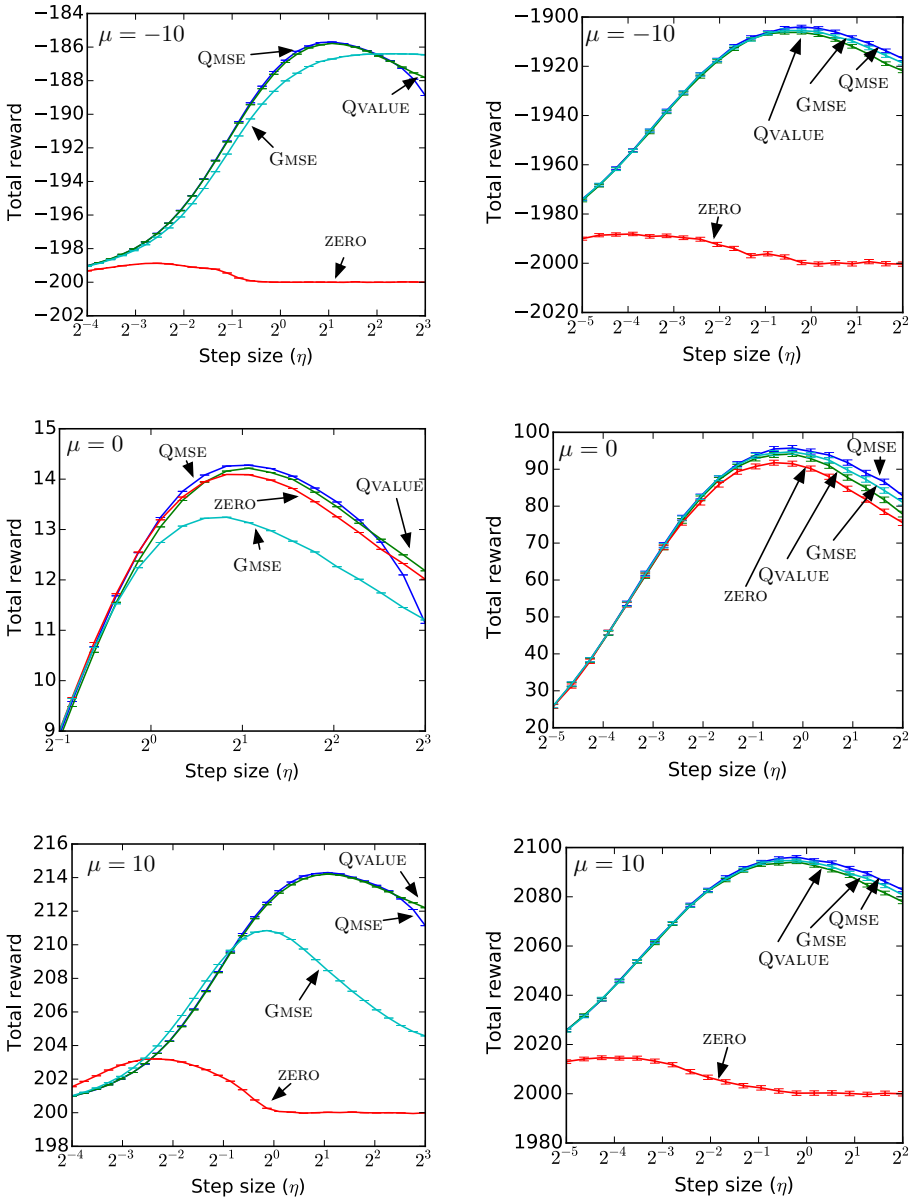

Figure 4.5: Comparisons of the four estimated baselines with good initializions in the ten-armed the triangle test-beds. The mean reward for each action is drawn from a Gaussian distribution with mean μ and unit standard deviation. In the first row, $\mu = 10$ in the second row, $\mu = 0$, and in the third row $\mu = -10$. The left column shows estimates of the expected total reward in the ten-armed test-bed and the second column shows the same for the triangle test-bed. Again, the error bars show 3 standard errors in the mean.

Chapter 5

Learning in MDPCRs

This chapter describes the second project that I worked on during my MSc, which focused on designing and analyzing efficient learning algorithms for loop-free episodic and uniformly ergodic MDPCRs which were introduced in Section 2.2. We propose three new algorithms: an algorithm for learning in loop-free episodic MDPCRs under instructive (full-information) feedback, where the agent observes the entire reward function r_t after each action, an algorithm for learning in loop-free episodic MDPCRs under evaluative (bandit) feedback, where the agent only observes the reward for the action she took, and an algorithm for learning in uniformly ergodic MDPCRs under instructive feedback. We believe that the algorithm for learning under instructive feedback in ergodic MDPCRs can be extended to learning under evaluative feedback, but the analysis proved to be quite challenging. In all cases, we assume that the rewards belong to the interval $[0, 1]$. The theoretical results for these three algorithms either improve or complement the results for existing algorithms and often hold even under weaker conditions. This comes at the cost of having increased, though still polynomial, computational complexity.

A common strategy in computing science is to reduce a problem that we would like to solve to a problem that we have already solved. The strategy of this project is to reduce the problem of learning in MDPCRs to the problem of online linear optimization. Reduction in this case means that if I have an

algorithm for online linear optimization with provable regret bounds, then I should be able to use that algorithm to achieve a similar regret bound while learning in an MDPCR. Section 3.3 presented online mirror ascent, which is an algorithm for online linear optimization with a good regret bound. The three algorithms proposed in this project are all instances of online mirror ascent applied to the problem of learning in MDPCRs by way of a reduction to online linear optimization.

In all cases considered in ths project, online mirror ascent cannot be implemented exactly. The update rule of online mirror ascent has two steps: first, we compute the unconstrained maximizer of an objective function that combines the goals of maximizing the most recent payout vector and not moving too far from the previous choice. Second, we project the unconstrained maximizer back onto the set of feasible solutions. In many cases, the unconstrained maximizer can be computed as a simple closed-form expression but the projection step is expensive and can only be solved approximately. A natural question is: how do these approximations impact the performance of online mirror ascent? The final result of this project is of independent interest and provides theoretical analysis for a natural approximate implementation of online mirror ascent.

5.1 Reductions to Online Linear Optimization

In this section, we reduce loop-free episodic MDPCRs and uniformly ergodic MDPCRs to online linear optimization. Recall that in online linear optimization, the agent chooses a sequence of points $w_1, \ldots, w_T$ from a convex set $K \subset \mathbb{R}^d$. Following the agent's choice of w_t, her environment chooses a payout vector r_t and she earns reward equal to $r_t^\top w_t$. Her choice of w_t should only depend on $w_{1:(t-1)}$ and $r_{1:(t-1)}$, while the environment's choice of r_t should depend only on $w_{1:t}$ and $r_{1:(t-1)}$. The agent's goal is to choose the sequence w_t so that her regret relative to the best-in-hindsight fixed point in K is small. That is, she wants to minimize

$$\mathcal{R}_T(w_{1:T}, r_{1:T}) = \sup_{w \in K} r_t^\top (w - w_t).$$

We only provide reductions for the instructive feedback setting, where the agent observes the entire reward function, since our algorithm for the evaluative feedback setting are derived from the instructive case by statistically estimating the reward function.

5.1.1 Reduction of Loop-Free Episodic MDPCRs

This subsection shows that learning in loop-free episodic MDPCRs can be reduced to online linear optimization. Recall that in a loop-free episodic MDPCR, the state space $\mathcal{X}$ is partitioned into L layers $\mathcal{X}_1, \ldots, \mathcal{X}_L$ and that each episode starts in $\mathcal{X}_1$, and moves through the layers in order until the agent reaches $\mathcal{X}_L$. As a consequence, every episode visits exactly one state from each layer. Since each state can be visited at most once in an episode, we consider the case where the reward function and the agent's policy only change at the end of an episode, rather than at every time step. We denote the reward function and the agent's policy for the τ^{th} episode by r_τ and π_τ, respectively.

The main idea behind the reduction from learning in loop-free MDPCRs to online linear optimization is to represent the agent's policies in such a way that the expected total reward in the τ^{th} episode is a linear function of the representation of the policy π_τ. With such a policy representation, we can construct an online linear optimization game where in each round the agent chooses a policy for episode τ and the linear payout vector for that round is set so that the agent's reward in the linear optimization round is exactly the expected total reward in the τ^{th} episode.

We will represent policies by their occupancy measure. The occupancy measure of a policy π in a loop-free episodic MDPCR describes how often an agent following π will visit each state. An agent following policy π will visit exactly one state in each layer $\mathcal{X}_\ell$ and, since the transitions in an MDPCR are stochastic, there is a well-defined probability of visiting each state $x \in \mathcal{X}_\ell$. The (state) occupancy measure of a policy π is the map $\nu(\pi) : \mathcal{X} \to [0, 1]$ defined by

$$\nu(x, \pi) = \mathbb{P}(X_\ell^\pi = x),$$

where $\ell \in \{1, \dots, L\}$ is the layer index such that $x \in \mathcal{X}_\ell$ and X_ℓ^π is the random state from layer $\mathcal{X}_\ell$ visited by the agent. In words, $\nu(x, \pi)$ is the probability that an agent following policy π will visit state x in an episode. The (state-action) occupancy measure of a policy, denote by $\mu(\pi)$, is defined by

$$\mu(x, a, \pi) = \mathbb{P}(X_\ell^\pi = x, A_\ell^\pi = a) = \nu(x, \pi)\pi(x, a).$$

The quantity $\mu(x, a, \pi)$ is the probability that an agent following policy π will be in state x and choose action a. For the rest of this section, ν will always refer to the state occupancy measure of a policy, and μ will always refer to the state-action occupancy measure.

Our plan is to represent policies by their state-action occupancy measures and to choose policies by playing an online linear optimization game over the set of state-action occupancy measures. For this approach to be sensible, we need to show that the state-action occupancy measures can actually be used to represent policies (i.e., all policies have one, and the policy can be determined from only the occupancy measure). In order to apply online mirror ascent, we need to show that the set of occupancy measures is a convex set. Finally, to make the connection between the online linear optimization game and learning in the MDPCR, we need to show that the expected total episodic reward is a linear function of the state action-occupancy measure.

First, we show that it is possible to recover a policy from its state-action occupancy measure.

Lemma 5.1. *Let $\mu : \mathcal{X} \times \mathcal{A} \to [0, 1]$ be the state-action occupancy measure of some unknown policy π. Set*

$$\hat{\pi}(x, a) = \begin{cases} \mu(x, a)/\nu(x) & \text{if } \nu(x) > 0 \\ 1/|\mathcal{A}| & \text{otherwise,} \end{cases}$$

where $\nu(x) = \sum_a \mu(x, a)$. Then $\hat{\pi}(x, a) = \pi(x, a)$ for all states x that π visits with non-zero probability.

Proof. Suppose that π is the unknown policy. Then we have $\mu(x, a) =$

$\mu(x, a, \pi)$ and $\nu(x) = \nu(x, \pi)$. From the definition of $\mu(x, a, \pi)$, for each state x we have

$$\sum_a \mu(x, a, \pi) = \sum_a \nu(x, \pi)\pi(x, a) = \nu(x, \pi).$$

Further, whenever $\nu(x, \pi) \neq 0$, we can divide the equation $\mu(x, a, \pi) = \nu(x, \pi)\pi(x, a)$ by $\nu(x, \pi)$ to obtain

$$\pi(x, a) = \frac{\mu(x, a, \pi)}{\nu(x, \pi)} = \frac{\mu(x, a, \pi)}{\sum_{a'} \mu(x, a', \pi)}.$$

It follows that $\hat{\pi}(x, a) = \pi(x, a)$ whenever $\nu(x) > 0$. □

This lemma shows that, given only the state-action occupancy measure of a policy, we can recover the policy's action-selection probabilities in every state that it visits with non-zero probability. It is not a serious problem that we cannot recover the action selection probabilities in the remaining states, since an agent following the policy will visit them with probability zero. Therefore, since every policy has a state-action occupancy measure and, since we can (essentially) recover a policy from any state-action occupancy measure, we are able to represent policies by their state-action occupancy measures. In the language of policy gradient methods, we can think of the map given by Lemma 5.1 as a policy parameterization.

Next, we want to show that the set of all state-action occupancy measures $K = \{\mu(\pi) : \pi \in \Pi\}$ is a convex subset of $\mathbb{R}^d$. Let $d = |\mathcal{X} \times \mathcal{A}|$ be the number of state-action pairs. Then we can think of the set of functions $\{f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}\}$ as a d-dimensional vector space by identifying functions with tables (or vectors) of their values at each of the d state-action pairs. In this space of functions the natural inner product is defined by $f^\top g = \sum_{x,a} f(x, a)g(x, a)$. For the rest of this chapter, we treat functions with finite domains as vectors in finite-dimensional vector spaces together with this inner product. With this convention, we can show that the set of occupancy measures is a convex set.

Lemma 5.2. *Fix a loop-free episodic MDPCR and let* $K = \{\mu(\pi) : \pi \in \Pi\} \subset$

$\mathbb{R}^d$ *be the set of occupation measures. Then*

$$K = \left\{ \mu : \mathcal{X} \times \mathcal{A} \to [0,1] \,:\, \nu(x_{\text{start}}) = 1, \forall x' \in \mathcal{X} : \nu(x') = \sum_{x,a} \mu(x,a)\mathcal{P}(x,a,x') \right\},$$

where we used the shorthand $\nu(x) = \sum_a \mu(x,a)$. *Moreover, since* K *is defined by a set of linear inequalities, it is a convex subset of* $\mathbb{R}^d$.

Finally, we want to show that the expected total reward of an agent following policy π_τ in the τ^{th} epsiode is a linear function of $\mu(\pi_\tau)$. We obtain this result by applying the following lemma with $f = r_\tau$.

Lemma 5.3. *Let* π *be any policy for a loop-free episodic MDPCR and let* $f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ *be any function. Then*

$$\mathbb{E}\left[\sum_{\ell=1}^{L} f(X_\ell^\pi, A_\ell^\pi)\right] = f^\top \mu(\pi).$$

Proof. The proof follows from the fact that for each state x in the ℓ^{th} layer, and for each action a, we have that $\mu(x,a,\pi) = \mathbb{P}(X_\ell^\pi = x, A_\ell^\pi = a)$.

$$\begin{aligned}
\mathbb{E}\left[\sum_{\ell=1}^{L} f(X_\ell^\pi, A_\ell^\pi)\right] &= \sum_{\ell=1}^{L} \mathbb{E}[f(X_\ell^\pi, A_\ell^\pi)] \\
&= \sum_{\ell=1}^{L} \sum_{x \in \mathcal{X}_\ell, a} \mathbb{P}(X_\ell^\pi = x, A_\ell^\pi = a) f(x,a) \\
&= \sum_{\ell=1}^{L} \sum_{x \in \mathcal{X}_\ell, a} \mu(x,a,\pi) f(x,a) \\
&= \sum_{x,a} \mu(x,a,\pi) f(x,a) \\
&= f^\top \mu(\pi).
\end{aligned}$$

□

Combining Lemmas 5.1, 5.2, and 5.3, we have the following reduction from learning in loop-free episodic MDPCRs to online linear optimization.

Theorem 5.4. *Let $M = (\mathcal{X}, \mathcal{A}, \mathcal{P}, (r_\tau)_{\tau \in \mathbb{N}})$ be a loop-free episodic MDPCR. Then, the regret of any sequence of policies $\pi_1, \ldots, \pi_T$ relative to the set of Markov policies is equal to the regret of the sequence of state-action occupation measures $\mu(\pi_1), \ldots, \mu(\pi_T)$ in an online linear optimization game where $K = \{\mu(\pi) : \pi \in \Pi\}$ and the adversary chooses the payout vector r_τ for the τ^{th} round to be equal to the reward function in the MDPCR for the τ^{th} episode.*

5.1.2 Reduction of Uniformly Ergodic MDPCRs

This subsection shows that learning in uniformly ergodic MDPCRs can be reduced to online linear optimization. Recall that in a uniformly ergodic MDPCR, every policy π has a unique stationary distribution $\nu(\pi) \in \Delta_{\mathcal{X}}$ and that each policy converges to its stationary distribution uniformly quickly. The stationary distribution over state-action pairs for a policy π, denoted by $\mu(\pi)$, is defined by

$$\mu(x, a, \pi) = \nu(x, \pi)\pi(x, a).$$

The reduction presented in this section is very similar to the reduction in the previous section with the state-action stationary distribution replacing the state-action occupancy measure.

In this case, we represent the agent's policies by their stationary distribution over the set of state-action pairs. Again, we need to show that it is possible to recover a policy from its stationary distribution, that the set of stationary distributions is convex, and that we can establish a relationship between the regret in an online linear optimization game and the regret in a uniformly ergodic MDPCR. In the loop-free episodic case, the relationship was very straight forward, while in this case the situation is slightly more subtle.

First, we show that it is possible to recover a policy π from its stationary distribution $\mu(\pi)$.

Lemma 5.5. *Let $\mu : \mathcal{X} \times \mathcal{A} \to [0, 1]$ be the state-action stationary distribu-*

tion of an unknown policy π. Set

$$\hat{\pi}(x,a) = \begin{cases} \mu(x,a)/\nu(x) & \text{if } \nu(x) > 0 \\ 1/|\mathcal{A}| & \text{otherwise,} \end{cases}$$

where $\nu(x) = \sum_a \mu(x,a)$. Then $\hat{\pi}(x,a) = \pi(x,a)$ for all states x with non-zero probability in the stationary distribution of π.

Proof. The proof is identical to the proof of Lemma 5.1. □

Next, we show that the set of stationary distributions is a convex subset of $\mathbb{R}^d$ when we identify the set of functions $\{f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}\}$ with tables (or vectors) of their values at each of the $d = |\mathcal{X} \times \mathcal{A}|$ state-action pairs.

Lemma 5.6. *Fix a uniformly ergodic MDPCR and let $K = \{\mu(\pi) : \pi \in \Pi\} \subset \mathbb{R}^d$ denote the set of stationary distributions. Then*

$$K = \left\{ \mu : \mathcal{X} \times \mathcal{A} \to [0,1] : \forall x' \in \mathcal{X} : \nu(x') = \sum_{x,a} \mu(x,a) P(x,a,x') \right\},$$

where we used the shorthand $\nu(x) = \sum_a \mu(x,a)$. Moreover, since K is defined by a set of linear inequalities, it is a convex subset of $\mathbb{R}^d$.

Finally, we want to show that the regret of an agent following policies $\pi_1, \ldots, \pi_T$ in the uniformly ergodic MDPCR can somehow be related to linear functions of $\mu(\pi_1), \ldots, \mu(\pi_T)$. In the loop-free episodic case, the expected reward in each episode was exactly the inner product $r_\tau^\top \mu(\pi_\tau)$. In the uniformly ergodic case, the inner product $r_t^\top \mu(\pi_t)$ is the long-term average reward of the policy π_t in the DP (not MDPCR) with deterministic rewards given by r_t and with the same states, actions, and transition probabilities as the MDPCR. The following lemma shows that we can bound the agent's regret in the MDPCR in terms linear functions of the stationary distributions.

Lemma 5.7. *Fix a uniformly ergodic MDPCR with mixing time $\tau < \infty$ and suppose that the reward functions satisfy $r_t(x,a) \in [0,1]$ for all times,*

states, and actions. Let $B > 0$ and suppose $\pi_1, \ldots, \pi_T$ are any sequence of policies with $\left\|\nu(\pi_{t-1}) - \nu(\pi_t)\right\|_1 \leq B$ for all times $t = 2, \ldots, T$. Then the regret of the sequence of policies $\pi_1, \ldots, \pi_T$ relative to any fixed policy π can be bounded as follows:

$$\mathbb{E}_{\pi}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] - \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] \leq \sum_{t=1}^{T} r_t^{\top}\big(\mu(\pi) - \mu(\pi_t)\big) + (\tau+1)TB + 4\tau + 4.$$

Proof. Recall that we use the notation $\nu(\pi)$ and $\mu(\pi)$ for the state and state-action stationary distributions of the policy π, respectively. We now introduce notation for the finite-time distributions.

Notation for following the policy $\boldsymbol{\pi}$: Let $\tilde{\nu}_t^{\pi}(x) = \mathbb{P}_{\pi}(X_t = x)$ be the probability that an agent following policy π visits state x at time t and let $\tilde{\mu}_t^{\pi}(x, a) = \mathbb{P}_{\pi}(X_t = x, A_t = a)$ be the probability that she takes action a from state x at time t. We have

$$\begin{aligned}\tilde{\mu}_t^{\pi}(x, a) &= \mathbb{P}_{\pi}(X_t = x, A_t = a) \\ &= \mathbb{P}_{\pi}(A_t = a \,|\, X_t = x)\mathbb{P}_{\pi}(X_t = x) \\ &= \pi(x, a)\tilde{\nu}_t^{\pi}(x).\end{aligned}$$

The following recursive expression for $\tilde{\nu}_t^{\pi}$ will be useful: Since the agent starts in the state x_{start} with probability one, we know that $\tilde{\nu}_1^{\pi}(x) = \mathbb{I}\{x = x_{\text{start}}\}$. For each $t \geq 1$, we have

$$\tilde{\nu}_{t+1}^{\pi} = \tilde{\nu}_t^{\pi} P^{\pi},$$

where P^{π} is as in Definition 2.15 and is an operator on $\Delta_{\mathcal{X}}$ that corresponds to taking a single step according to the policy π.

Notation for following the sequence of policies $\boldsymbol{\pi_{1:T}}$: Similarly, let $\tilde{\nu}_t(x) = \mathbb{P}_{\pi_{1:T}}(X_t = x)$ be the probability that an agent following the sequence of policies $\pi_{1:T}$ visits state x at time t and let $\tilde{\mu}_t(x, a) = \mathbb{P}_{\pi_{1:T}}(X_t =$

$x, A_t = a)$ be the probability that she takes action a from state x. Again, we have $\tilde{\mu}_t(x,a) = \pi_t(x,a)\tilde{\nu}_t(x)$, and we can express $\tilde{\nu}_t$ recursively: $\tilde{\nu}_1(x) = \mathbb{I}\{x = x_{\text{start}}\}$ and

$$\tilde{\nu}_{t+1} = \tilde{\nu}_t P^{\pi_t}.$$

With the above notation, we are ready to prove the lemma. First, we rewrite the two expectations as sums:

$$\begin{aligned}
\mathbb{E}_\pi\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] &= \sum_{t=1}^{T} \mathbb{E}_\pi\big[r_t(X_t, A_t)\big] \\
&= \sum_{t=1}^{T} \sum_{x,a} \tilde{\mu}_t^\pi(x,a) r_t(x,a) \\
&= \sum_{t=1}^{T} r_t^\top \tilde{\mu}_t^\pi.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] &= \sum_{t=1}^{T} \mathbb{E}_{\pi_{1:T}}\big[r_t(X_t, A_t)\big] \\
&= \sum_{t=1}^{T} \sum_{x,a} \tilde{\mu}_t(x,a) r_t(x,a) \\
&= \sum_{t=1}^{T} r_t^\top \tilde{\mu}_t.
\end{aligned}$$

With this, the expected regret of the policies $\pi_1, \ldots, \pi_T$ relative to the fixed policy π can be written as:

$$\mathbb{E}_\pi\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] - \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] = \sum_{t=1}^{T} r_t^\top (\tilde{\mu}_t^\pi - \tilde{\mu}_t).$$

We can add and subtract the stationary distributions of π and π_t into the

t^{th} term of the sum above to obtain the following decomposition:

$$\sum_{t=1}^{T} r_t^\top(\tilde{\mu}_t^\pi - \tilde{\mu}_t) = \sum_{t=1}^{T} r_t^\top(\tilde{\mu}_t^\pi - \mu(\pi)) + \sum_{t=1}^{T} r_t^\top(\mu(\pi) - \mu(\pi_t)) + \sum_{t=1}^{T} r_t^\top(\mu(\pi_t) - \tilde{\mu}_t). \tag{5.1}$$

The middle term of the above decomposition appears in the bound from the statement of the lemma, so it remains to upper bound the first and last term by $(\tau+1)BT + 4\tau + 4$.

To bound the first term we use the following lemma from [NGSA2014]

Lemma 5.8 (Lemma 1 from [NGSA2014])**.**

$$\sum_{t=1}^{T} r_t^\top(\tilde{\mu}_t^\pi - \mu(\pi)) \leq 2\tau + 2. \tag{5.2}$$

To bound the last term, we use the following technical lemma:

Lemma 5.9. *For each time t, we have*

$$\begin{aligned}\left\|\tilde{\nu}_t - \nu(\pi_t)\right\|_1 &\leq 2e^{-(t-1)/\tau} + B\sum_{s=0}^{t-2} e^{-s/\tau} \\ &\leq 2e^{-(t-1)/\tau} + B(\tau+1).\end{aligned}$$

Moreover, for each time t, we have

$$\left\|\tilde{\mu}_t - \mu(\pi_t)\right\|_1 \leq 2e^{-(t-1)/\tau} + B(\tau+1).$$

Proof. We prove the first inequality by induction on t. The base case is when $t = 1$. By the triangle inequality and the fact that $\tilde{\nu}_1$ and $\nu(\pi_1)$ are distributions, we have $\left\|\tilde{\nu}_1 - \nu(\pi_1)\right\|_1 \leq \left\|\tilde{\nu}_1\right\|_1 + \left\|\nu(\pi_1)\right\|_1 \leq 2 = 2e^{-(1-1)/\tau} + B\sum_{s=0}^{-1} e^{-s/\tau}$. Therefore, the claim holds when $t = 1$. Now suppose that

the claim holds for t. Then we have

$$\begin{aligned}
\left\|\tilde{\nu}_{t+1} - \nu(\pi_{t+1})\right\|_1 &\leq \left\|\tilde{\nu}_{t+1} - \nu(\pi_t)\right\|_1 + \left\|\nu(\pi_t) - \nu(\pi_{t+1})\right\|_1 && \text{(Triangle Inequality)} \\
&\leq \left\|\tilde{\nu}_t P^{\pi_t} - \nu(\pi_t) P^{\pi_t}\right\|_1 + B && \text{(Stationarity of } \nu(\pi_t)\text{)} \\
&\leq e^{-1/\tau} \left\|\tilde{\nu}_t - \nu(\pi_t)\right\|_1 + B && \text{(Uniformly Ergodic)} \\
&\leq e^{-1/\tau} \left(2e^{-(t-1)/\tau} + B \sum_{s=0}^{t-2} e^{-s/\tau} \right) + B && \text{(Induction Hypothesis)} \\
&= 2e^{-(t+1-1)/\tau} + B \sum_{s=0}^{t+1-2} e^{-s/\tau}.
\end{aligned}$$

It follows that the first inequality holds for all times t. The second inequality follows from the first, together with the fact that

$$\sum_{s=0}^{t-2} e^{-s/\tau} \leq 1 + \int_0^\infty e^{-s/\tau}\, ds = 1 + \tau.$$

The final inequality is proved as follows:

$$\begin{aligned}
\left\|\tilde{\mu}_t - \mu(\pi_t)\right\|_1 &= \sum_{x,a} \left|\tilde{\mu}_t(x,a) - \mu(x,a,\pi_t)\right| \\
&= \sum_{x,a} \left|\tilde{\nu}_t(x)\pi_t(x,a) - \nu(x,\pi)\pi_t(x,a)\right| \\
&= \sum_{x} \left|\tilde{\nu}_t(x) - \nu(x,\pi_t)\right| \sum_{a} \pi_t(x,a) \\
&= \sum_{x} \left|\tilde{\nu}_t(x) - \nu(x,\pi_t)\right| \\
&= \left\|\tilde{\nu}_t - \nu(\pi_t)\right\|_1 \\
&\leq 2e^{-(t-1)/\tau} + B(\tau + 1).
\end{aligned}$$

□

We are finally ready to bound the sum $\sum_{t=1}^{T} r_t^\top(\tilde{\mu}_t - \mu(\pi_t))$.

$$\begin{aligned}
\sum_{t=1}^{T} r_t^\top(\tilde{\mu}_t - \mu(\pi_t)) &\le \sum_{t=1}^{T} \|r_t\|_\infty \|\tilde{\mu}_t - \mu(\pi_t)\|_1 && \text{(Hölder's Inequality)} \\
&\le \sum_{t=1}^{T} \left(2e^{-(t-1)/\tau} + B(\tau+1)\right) && \text{(Lemma 5.9, } \|r_t\|_\infty \le 1) \\
&= (\tau+1)BT + 2\sum_{t=1}^{T} e^{-(t-1)/\tau} \\
&\le (\tau+1)BT + 2\tau + 2, && (5.3)
\end{aligned}$$

where in the last line we again used $\sum_{t=1}^{T} e^{-(t-1)/\tau} \le \tau + 1$.

Substituting inequalities (5.2) and (5.3) into the regret decomposition (5.1) proves the lemma. □

Combining Lemmas 5.5, 5.6 and 5.7 gives the following reduction from learning in uniformly ergodic MDPCRs to online linear optimization.

Theorem 5.10. *Fix a uniformly ergodic MDPCR with mixing time $\tau < \infty$ and bounded rewards: $r_t(x,a) \in [0,1]$ for all states, actions, and times. Consider the online linear optimization problem where K is the set of stationary distributions over state-action pairs and the environment chooses the payout vector in round t to be equal to the t^{th} MDPCR reward function. Suppose an agent for the online linear optimization game chooses the stationary distributions $\mu(\pi_1), \ldots, \mu(\pi_T)$. Then we can essentially recover the policies $\pi_1, \ldots, \pi_T$, and if an agent follows those policies in the MDPCR, then her regret is bounded by*

$$\begin{aligned}
&\mathbb{E}_\pi\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] - \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] \\
&\qquad \le \sum_{t=1}^{T} r_t^\top\big(\mu(\pi) - \mu(\pi_t)\big) + (\tau+1)TB + 4\tau + 4
\end{aligned}$$

for any B such that $B \ge \|\nu(\pi_{t-1}) - \nu(\pi_t)\|_1$ for all $t = 2, \ldots, T$.

This reduction shows that if we can achieve low regret in an online linear optimization problem, and if the sequence of choices don't change too quickly, then we can also achieve low regret in a uniformly ergodic MDPCR.

5.2 Online Mirror Ascent with Approximate Projections

A natural idea is to use online mirror ascent to learn low-regret sequences of policies for MDPCRs by way of their reduction to online linear optimization. Recall that online mirror ascent update has two steps: first we compute an update that is not constrained to the set K which we then project back onto K. In most cases, the unconstrained update can be expressed as closed-form expression which is efficiently evaluatable. When the set K is simple (such as the unit ball or the probability simplex) and the Bregman divergence is chosen appropriately, the projection step may also have a closed-form expression that can be evaluated efficiently. In general, however, computing the Bregman projection onto the set K is a convex optimization problem whose solution must be approximated iteratively by, for example, interior point optimization methods. This section addresses the important question: How much additional regret is incurred by using approximate projections? To the best of our knowledge, this is the first formal analysis of online mirror ascent with approximate projections, despite the fact that in most applications the projection step must be approximated.

Formally, we consider the following notion of approximate projection: Fix any constant $c > 0$. Let $R : S \to \mathbb{R}$ be a σ-strongly convex function with respect to the norm $\|\cdot\|$ and let K be a convex subset of S. For any point $w \in S$, we say that a point $w' \in K$ is a c-approximate projection of w onto K with respect to the Bregman divergence D_R if $\left\|w' - w^*\right\| < c$ where $w^* = \operatorname{argmin}_{u \in K} D_R(u, w)$ is the exact projection. Algorithm 8 gives pseudocode for online mirror ascent with c-approximate projections.

Theorem 5.11. *Let $R : S \to \mathbb{R}$ be a convex Legendre function and $K \subset S$ be a convex set such that R is L-Lipschitz on K wrt $\|\cdot\|$ (that is, $\|\nabla R(u) - \nabla R(w)\| \leq$*

Input: Step size $\eta > 0$, Regularizer $R : S \to \mathbb{R}$, $S \supset K$, and a black-box P_K that computes c-approximate projections onto K

Choose $w_1 \in K$ arbitrarily;
for *each round* $t = 1, 2, \ldots$ **do**
  Optionally use w_t in some other computation;
  Set $w_{t+1/2} = \operatorname{argmin}_{u \in S} \eta r_t^\top u + D_R(u, w_t)$;
  Set $w_{t+1} = P_K(w_{t+1/2})$;
end

Algorithm 8: Online Mirror Ascent with c-approximate Projections

$L \cdot \|u - w\|$ for all $u, w \in K$). Let $D = \sup_{u,v \in K} \|u - v\|_*$ be the diameter of K with respect to the dual norm of $\|\cdot\|$. Then the regret of online mirror ascent with c-approximate projections, step size $\eta > 0$ and regularizer R satisfies

$$\sum_{t=1}^{T} r_t^\top (w - w_t) \leq \frac{D_R(w, w_1)}{\eta} + \frac{cLDT}{\eta} + \sum_{t=1}^{T} r_t^\top (w_{t+1/2} - w_t).$$

Moreover, when $c = 0$ the claim holds even when $L = \infty$.

Proof. This roughly follows the proof of Theorem 15.4 from [GPS2014] with the appropriate modifications to handle the case where the projections are only c-approximate.

Let $w_1, \ldots, w_T \in K$ be the sequence of points generated by online mirror ascent with c-approximate projections, let $w_{t+1/2}$ be the unprojected updates for $t = 1, \ldots, T$, and, finally, let $w^*_{t+1} = \operatorname{argmin}_{u \in K} D_R(u, w_{t+1/2})$ be the exact projection of $w_{t+1/2}$ onto the set K.

For each $t = 1, \ldots, T$, we know that $w_{t+1/2}$ is the unconstrained minimizer of the objective function $J_t(u) = \eta r_t^\top u - D_R(u, w_t)$ and therefore

$\nabla J_t(w_{t+1/2}) = 0$. We can compute the gradient of J_t to be

$$\begin{aligned}\nabla J(u) &= \nabla\big[\eta r_t^\top u - R(u) + R(w_t) + \nabla R(w_t)^\top (u - w)\big] \\ &= \eta r_t - \nabla R(u) + \nabla R(w_t).\end{aligned}$$

Rearranging the condition $\nabla J(w_{t+1/2}) = 0$ gives

$$r_t = \frac{1}{\eta}\big(\nabla R(w_{t+1/2}) - \nabla R(w_t)\big).$$

Therefore, for each time t we have

$$\begin{aligned}r_t^\top (w - w_t) &= \frac{1}{\eta}\big(\nabla R(w_{t+1/2}) - \nabla R(w_t)\big)^\top (w - w_t) \\ &= \frac{1}{\eta}\big(D_R(w, w_t) - D_R(w, w_{t+1/2}) + D_R(w_t, w_{t+1/2})\big),\end{aligned}$$

where the second line is obtained by a long but straight-forward calculation.

From the Pythagorean theorem for Bregman divergences, we have that

$$D_R(w, w_{t+1/2}) \geq D_R(w, w^*_{t+1}) - D_R(w^*_{t+1}, w) \geq D_R(w, w^*_{t+1}).$$

Substituting this above gives

$$\begin{aligned}r_t^\top (w - w_t) &\leq \frac{1}{\eta}\big(D_R(w, w_t) - D_R(w, w^*_{t+1}) + D_R(w_t, w_{t+1/2})\big) \\ &= \frac{1}{\eta}\big(D_R(w, w_t) - D_R(w, w_{t+1}) + D_R(w, w_{t+1}) - D_R(w, w^*_{t+1}) \\ &\qquad + D_R(w_t, w_{t+1/2})\big).\end{aligned}$$

Summing from $t = 1$ to T, The first two terms of the above expression will

telescope, leaving only the first and last:

$$\begin{aligned}\sum_{t=1}^{T} r_t^\top (w - w_t) &\le \frac{D_R(w, w_1)}{\eta} - \frac{D_R(w, w_{T+1})}{\eta} + \frac{1}{\eta}\sum_{t=1}^{T} D_R(w_t, w_{t+1/2}) \\ &\quad + \frac{1}{\eta}\sum_{t=1}^{T}\big(D_R(w, w_{t+1}) - D_R(w, w^*_{t+1})\big) \\ &\le \frac{D_R(w, w_1)}{\eta} + \frac{1}{\eta}\sum_{t=1}^{T} D_R(w_t, w_{t+1/2}) \\ &\quad + \frac{1}{\eta}\sum_{t=1}^{T}\big(D_R(w, w_{t+1}) - D_R(w, w^*_{t+1})\big). \qquad (5.4)\end{aligned}$$

All that remains is to bound the two sums in (5.4).

We can bound the first sum as follows: since Bregman divergences are non-negative, we have

$$\begin{aligned}D_R(w_t, w_{t+1/2}) &\le D_R(w_t, w_{t+1/2}) + D_R(w_{t+1/2}, w_t) \\ &= \big(\nabla R(w_t) - \nabla R(w_{t+1/2})\big)^\top (w_t - w_{t+1/2}) \\ &= \eta r_t^\top (w_{t+1/2} - w_t).\end{aligned}$$

Substituting this into the first sum gives

$$\frac{1}{\eta}\sum_{t=1}^{T} D_R(w_t, w_{t+1/2}) \le \sum_{t=1}^{T} r_t^\top (w_{t+1/2} - w_t).$$

We can bound the second sum as follows: First, if $c = 0$ then $w_t = w^*_t$ and the sum is zero. In this case, we never needed the condition that ∇R was L-Lipschitz. If $c > 0$, then, since R is a convex function, we have that

$$R(w_t) \ge R(w^*_t) + \nabla R(w^*_t)^\top (w - w^*_t).$$

Rearranging this inequality gives

$$R(w^*_t) - R(w_t) \le \nabla R(w^*_t)^\top (w^*_t - w_t).$$

Expanding $D_r(w, w_t) - D_R(w, w_t^*)$ and using the above inequality gives

$$\begin{aligned} D_R(w, w_t) - D_R(w, w_t^*) &= R(w_t^*) - R(w_t) + \nabla R(w_t^*)^\top (w - w_t^*) - \nabla R(w_t)^\top (w - w_t) \\ &\leq \nabla R(w_t^*)^\top (w - w_t) - \nabla R(w_t)^T (w - w_t) \\ &= \left(\nabla R(w_t^*) - \nabla R(w_t)\right)^\top (w - w_t) \\ &\leq \|\nabla R(w_t^*) - \nabla R(w_t)\| \, \|w - w_t\|_* \\ &\leq L \, \|w_t^* - w_t\| \, D \\ &\leq cLD. \end{aligned}$$

Finally, substituting this into the second sum gives

$$\frac{1}{\eta} \sum_{t=1}^{T} D_R(w, w_t) - D_R(w, w_t^*) \leq \frac{cLDT}{\eta}$$

Substituting the above bounds into (5.4) completes the proof. □

When the regularizer R is σ-strongly convex wrt $\|\cdot\|$, we can use the following lemma to bound the sum $\sum_t r_t^\top (w_t - w_{t+1/2})$ in Theorem 5.11.

Lemma 5.12. *Let $R : S \to \mathbb{R}$ be a σ-strongly convex Legendre function wrt the norm $\|\cdot\|$, $\eta > 0$, $w_t \in S$, $r_t \in \mathbb{R}^d$ and defined $w_{t+1/2}$ to be the unconstrained mirror ascent update $w_{t+1/2} = \operatorname{argmin}_{u \in S} \eta r_t^\top u + D_R(u, w_t)$. Then*

$$r_t^\top (w_{t+1/2} - w_t) \leq \frac{\eta}{\sigma} \|r_t\|_*^2 ,$$

where $\|\cdot\|_*$ denotes the dual norm of $\|\cdot\|$.

Proof. As in the proof of Theorem 5.11, we have that $r_t = \frac{1}{\eta}(\nabla R(w_{t+1/2}) - \nabla R(w_t))$. Since R is σ-strongly convex, for all u and v in K, we have

$$R(u) \geq R(v) + \nabla R(v)^\top (u - v) + \frac{\sigma}{2} \|u - v\|^2 .$$

Summing and rearranging two instances of this inequality, one with $u = w_t$

and $v = w_{t+1/2}$, and one with $u = w_{t+1/2}$ and $v = w_t$, gives

$$\begin{aligned}\left\|w_t - w_{t+1/2}\right\|^2 &\leq \frac{1}{\sigma}\left(\nabla R(w_{t+1} - \nabla R_w\right)^\top (w_t - w_{t+1/2}) \\ &\leq \frac{1}{\sigma}\left\|\nabla R(w_{t+1/2}) - \nabla R(w_t)\right\|_* \left\|w_t - w_{t+1/2}\right\| \\ &= \frac{\eta}{\sigma}\left\|r_t\right\|_* \left\|w_t - w_{t+1}\right\|.\end{aligned}$$

Dividing both sides by $\|w_t - w_{t+1}\|$ shows that $\left\|w_t - w_{t+1/2}\right\| \leq \frac{\eta}{\sigma}\left\|r_t\right\|_*$. Therefore,

$$r_t^\top (w_{t+1/2} - w_t) \leq \left\|r_t\right\|_* \left\|w_{t+1/2} - w_t\right\| \leq \frac{\eta}{\sigma}\left\|r_t\right\|_*^2,$$

completing the proof. □

5.3 Learning Algorithms and Regret Bounds

This section introduces three new learning algorithms for MDPCRs. All three algorithms use online mirror ascent with approximate projections to choose a sequence of occupancy measures / stationary distributions in the online linear optimization problems from Section 5.1. All of these algorithms have the same interpretation: On each time step (or epsiode), the agent observes the rewards in some or all of the states. Following this observation, the agent updates her policy so that the occupancy measure / stationary distribution places more weight on the states that had high rewards. Intuitively, the agent makes a small update to her policy so that she spends more time taking actions from states which give high rewards.

In order to get the best regret bounds, we should choose the regularizer function R for online mirror ascent so that the induced Bregman divergence matches the geometry of the underlying problem. In the uniformly ergodic MDPCR case, the set K consists of probability distributions, and it is natural to measure distances between them in terms of the Kullback-Leibler (KL) divergence. Recall that we identify the set of functions $\{f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}\}$ as a $d = |\mathcal{X} \times \mathcal{A}|$-dimensional vector space. Consider the regularizer $J : (0, \infty)^d$

defined by

$$J(w) = \sum_{x,a} \big(w(x,a) \ln(w(x,a)) - w(x,a) \big).$$

This is the so-called unnormalized negative entropy (negentropy) regularizer, and the induced Bregman divergence is

$$D_J(u,w) = \sum_{x,a} \left(u(x,a) \ln\left(\frac{u(x,a)}{w(x,a)} \right) + w(x,a) - u(x,a) \right),$$

which is the unnormalized relative entropy between the non-negative functions u and w. When u and w are probability vectors (i.e., their components sum to one), then $D_J(u,w)$ is exactly the KL divergence between the vectors.

Similarly, in the loop free episodic case, the set K consists of occupancy measures, which are distributions when restricted to each of the layers $\mathcal{X}_1, \ldots, \mathcal{X}_L$ of the MDPCR. In this case, a natural choice is to choose the regularizer so that the induced Bregman divergence is the sum of the KL-divergences between the probability distributions on each layer. The unnormalized negentropy regularizer again accomplishes this.

For the regularizer J, the unconstrained update step of online mirror ascent is defined by

$$w_{t+1/2} = \operatorname*{argmax}_{u \in (0,\infty)^d} \eta r_t^\top u - \sum_{x,a} \left(u(x,a) \ln\left(\frac{u(x,a)}{w(x,a)} \right) + w(x,a) - u(x,a) \right).$$

This is a concave function of u, so we can find the maximizer by taking the derivative and setting it to zero, which yields

$$w_{t+1/2}(x,a) = w_t(x,a) \exp^{\eta r_t(x,a)}$$

for each state x and action a.

Moreover, suppose that the set $K \subset \big\{ w \in (0,\infty)^d : \|w\|_1 \leq B \big\}$. Then we have that R is $1/B$-strongly convex with respect to $\|\cdot\|_1$ on the set K.

Lemma 5.13 (Example 2.5 from [S2012]). *The function*

$$R(w) = \sum_{i=1}^{d} w(i)\log(w(i)) - w(i)$$

is $1/B$*-strongly convex with respect to* $\|\cdot\|_1$ *over the set*

$$S = \left\{ w \in \mathbb{R}^d \,:\, w_i > 0, \|w\|_1 \leq B \right\}.$$

There are two problems with using the unnormalized negentropy regularizer, both coming from the fact that that ∇J is not Lipschitz continuous on the set of occupancy measures or the set of stationary distributions. To see this, note that the partial derivatives of J are given by

$$\frac{\partial}{\partial w(x,a)} J(w) = \ln(w(x,a)),$$

which goes to $-\infty$ as $w(x,a)$ goes to 0. In general, there will be policies that have occupancy measures or stationary distributions with components equal to zero, which means the gradients of J will be unbounded. This prevents us from applying the results from Theorem 5.11 and makes it challenging to compute c-approximate projections. In each case, we deal with this by approximating K with the slightly smaller set $K_\alpha = \{\mu \in K \,:\, \forall x, a \,:\, \mu(x,a) \geq \alpha\}$ which contains only occupancy measures or stationary distributions that put at least mass α on every state-action pair. We will be able to use online mirror ascent with approximate projections to choose occupancy measures / stationary distributions from the set K_α which have low-regret relative to the best in K_α, and we will show that the best policy in K can't be much better than the best policy in K_α, which gives us a regret bound relative to the entire set of Markov policies. The restricted set K_α forces the agent to choose policies that explore the state and action spaces sufficiently well. In the evaluative feedback setting, where the agent must explore to find good actions, this would be a good idea even if the theory did not require it.

The following subsections give detailed descriptions of the three algo-

rithms and corresponding regret bounds.

5.3.1 Loop Free Episodic MDPCRs with Instructive Feedback

This section introduces an algorithm for learning in loop free episodic MDPCRs under instructive feedback, where the agent observes the entire reward function r_t after choosing her action at time t. For the remainder of this section, fix a loop-free episodic MDPCR $M = (\mathcal{X}, \mathcal{A}, \mathcal{P}, (r_t)_{t\in\mathbb{N}})$ with layers $\mathcal{X}_1, \ldots, \mathcal{X}_L$ and $r_\tau(x, a) \in [0, 1]$ for all states, actions, and episode indices. Let $d = |\mathcal{X} \times \mathcal{A}|$ be the number of state action pairs and set $K \subset \mathbb{R}^d$ to be the convex set of occupancy measures described by Lemma 5.2. Finally, let $\beta > 0$ be such that there exists an exploration policy π_{exp} with $\mu(x, a, \pi_{\text{exp}}) \geq \beta$ for all states x and actions a. This guarantees that for all $\alpha < \beta$, the set K_α is non-empty.

Algorithm 9 gives pseudocode for the proposed method and Theorem 5.14 applies the lemmas from the previous section to get a regret bound.

Input: Step size $\eta > 0$, exploration constant $\delta \in (0, 1]$, approximation constant $c > 0$

Choose $\mu_1 \in K_{\delta\beta}$ arbitrarily;

for *Each episode index* $\tau = 1, \ldots, T$ **do**

  Execute one episode following policy π_τ, obtained from μ_τ according to Lemma 5.1;

  Receive complete reward function r_τ from environment;

  Set $\mu_{\tau+1/2}(x, a) = \mu_\tau(x, a) \exp(\eta r_\tau(x, a))$ for each state x and action a;

  Set $\mu_{t+1} = P_{K_{\delta\beta}}(\mu_{t+1/2})$, where $P_{K_{\delta\beta}}$ is a black-box that computes c-approximate projections onto $K_{\delta\beta}$ wrt $\|\cdot\|_1$.

end

Algorithm 9: Approximate Online Mirror Ascent for Loop Free Episodic MDPCRs Under Instructive Feedback

Theorem 5.14. *Let M be a loop free episodic MDPCR, $\pi \in \Pi$ be any Markov policy, and $\pi_1, \ldots, \pi_T$ be the sequence of policies produced by Algorithm 9 with parameters $\delta \in (0,1]$, $c = \frac{\beta\delta\eta}{\sqrt{T}}$, and $\eta = \sqrt{\frac{D_{\max}}{LT}}$ where L is the number of layers in the MDPCR and $D_{\max} \geq \sup_{\mu \in K_{\delta\beta}} D_J(\mu, \mu(\pi_1))$. Then, the regret of an agent that follows the sequence of policies $\pi_{1:T}$ relative to the fixed policy π is bounded as follows*

$$\mathbb{E}_{\pi}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] - \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_\tau(X_t, A_t)\right] \leq 2\sqrt{LTD_{\max}} + \sqrt{T} + L\delta T,$$

and the per-time-step computational cost is $O(H(\beta\delta, c) + d)$, where $H(\beta\delta, c)$ is the cost of the black-box approximate projection routine and d is the number of state-action pairs.

Proof. First, we show that the agen't does not incur too much additional regret by choosing policies from the set $K_{\delta\beta}$ rather than the larger set K. For any occupancy measure $\mu \in K$, consider the mixed measure $\mu_\delta = (1-\delta)\mu + \delta\mu(\pi_{\exp})$. We have the following properties: $\mu_\delta(x,a) = (1-\delta)\mu(x,a) + \delta\mu(x,a,\pi_{\exp}) \geq \delta\beta$, and therefore $\mu_\delta \in K_{\delta\beta}$. Second, for any payout vector r with $r(x,a) \in [0,1]$, we have $|r^\top(\mu - \mu_\delta)| = \delta|r^\top(\mu(x,a) - \mu(x,a,\pi_{\exp})| \leq \delta\sum_{x,a} \leq \delta L$. Therefore, for any occupancy measure $\mu \in K$, there is an occupancy measure in $K_{\delta\beta}$ that earns nearly as much reward. This implies that having a good regret bound relative to any point in $K_{\delta\beta}$ gives us a regret bound relative to any point in K.

Next, we show that the regularizer J is $1/L$-strongly convex with respect to $\|\cdot\|_1$ on the set K. Since each occupancy measure μ is a distribution when restricted to the states and actions in a single layer $\mathcal{X}_\ell \times \mathcal{A}$, we have the following:

$$\|\mu\|_1 = \sum_{x,a} |\mu(x,a)| = \sum_{\ell=1}^{L} \sum_{x \in \mathcal{X}_\ell, a} |\mu(x,a)| = \sum_{\ell=1}^{L} 1 = L.$$

Therefore, by Lemma 5.13, we have that R is $1/L$-strongly convex on K with respect to $\|\cdot\|_1$.

Finally, we show that ∇J is Lipschitz continuous on $K_{\delta\beta}$ with respect to $\|\cdot\|_1$. Let $w \in K_{\delta\beta}$ be any occupancy measure and consider indices $i, j \in \{1, \ldots, d\}$ (in this proof, it is more convenient to use integer indices, rather than pairs of states and actions). Then we can compute the partial derivatives of $J(w)$ to be

$$\frac{\partial}{\partial w(i)} J(w) = \ln(w(i))$$
$$\frac{\partial^2}{\partial w(i)\partial w(j)} J(w) = \frac{\mathbb{I}\{i = j\}}{w(i)}.$$

It follows that the hessian $\nabla^2 J(w) \preceq I(\delta\beta)^{-1}$ and therefore ∇J is $\frac{1}{\delta\beta}$ Lipschitz continuous.

Let $\mu \in K$ be an arbirtary occupancy measure and let $\mu_\delta = (1-\delta)\mu + \delta\mu(\pi_{\exp})$ be the mixture of μ with the occupancy measure. Since J is $1/L$-strongly convex and ∇J is $\frac{1}{\delta\beta}$-Lipschitz on $K_{\delta\beta}$ we can apply Theorem 5.11 and Lemma 5.12 to get the following bound:

$$\begin{aligned}\sum_{\tau=1}^{T} r_\tau^\top(\mu_\delta - \mu(\pi_\tau)) &\le \frac{D_R(\mu_\delta, \mu(\pi_1))}{\eta} + \frac{cLDT}{\eta} + L\eta \sum_{\tau=1}^{T} \|r_\tau\|_\infty^* \\ &\le \frac{D_R(\mu_\delta, \mu(\pi_1))}{\eta} + \sqrt{T} + LT.\end{aligned}$$

Finally, since $r_\tau^\top(\mu - \mu_\delta) \le \delta L$, we have that

$$\begin{aligned}\sum_{\tau=1}^{T} r_\tau^\top(\mu - \mu(\pi_\tau)) &= \sum_{\tau=1}^{T} r_\tau^\top(\mu - \mu_\delta) + \sum_{\tau=1}^{T} r_\tau^\top(\mu_\delta - \mu(\pi_\tau)) \\ &\le \frac{D_{\max}}{\eta} + \sqrt{T} + \eta LT + LT\delta \\ &= 2\sqrt{TD_{\max}L} + \sqrt{T} + LT\delta.\end{aligned}$$

By Theorem 5.4, the same regret bounds holds for the sequence of policies in the MDPCR. □

Note that $D_{\max} = \Theta(L \ln \frac{1}{\pi_0})$, where $\pi_0 = \min_{(x,a)} \pi_{\exp}(x, a)$ (notice that

$\pi_{\exp}(x,a) \geq \beta$, since $\mu(x,a,\pi_{\exp}) \geq \beta$). If, for example, $\pi_{\exp}(x,\cdot)$ is selected to be the uniform distribution over $\mathcal{A}$, then $\beta > 0$ and $\pi_0 = 1/|\mathcal{A}|$, making the regret scale with $O(L\sqrt{T\ln(|\mathcal{A}|)})$ when $\delta = 1/\sqrt{T}$. Also, this makes the computational cost $\tilde{O}(d^{d.5}T^{1/4}/\sqrt{\beta})$, where $\tilde{O}$ hides log-factors. Neu *et al.* [NGS2010] gave an algorithm that achieves $O(L^2\sqrt{T\ln(|\mathcal{A}|)})$ regret with $O(d)$ computational complexity per time-step. Thus, our regret bound scales better in the problem parameters than that of Neu *et al.* [NGS2010], at the price of increasing the computational complexity. It is an interesting (and probably challenging) problem to achieve the best of the two results.

5.3.2 Loop Free Episodic MDPCRs with Evaluative Feedback

This section introduces an algorithm for learning in loop free episodic MDPCRs under evaluative feedback, where the agent only observes the reward $r_t(X_t, A_t)$ for the state X_t and action A_t that she visited during the t^{th} time step. The algorithm for this setting is essentially identical to the algorithm from the previous section for the instructive feedback setting, except we use importance sampling to estimate the complete reward function. Specifically, following the τ^{th} episode, we set

$$\hat{r}_\tau(x,a) = \begin{cases} \frac{r_\tau(x,a)}{\mu(x,a,\pi_\tau)} & \text{if } (x,a) \text{ was visited in the } \tau^{\text{th}} \text{ episode} \\ 0 & \textit{otherwise.} \end{cases} \tag{5.5}$$

and perform the update with this estimate of the complete reward function. Since $\hat{r}_\tau$ is only non-zero for state-action pairs visited by the agent in episode τ, it is known to the agent. This estimate is only well defined if $\mu(x,a,\pi_\tau) > 0$ for all states and actions, but since we restrict our algorithm to the set of occupancy measures which are lower bounded, this will always be the case for policies chosen by the algorithm. This particular reward approximation will be justified in the proof of Theorem 5.15.

As in the previous section, fix a loop free episodic MDPCR $\mathcal{M}$ with layers $\mathcal{X}_1, \ldots, \mathcal{X}_L$ and $r_\tau(x,a) \in [0,1]$ for all states, actions, and episode indices. Let $d = |\mathcal{X} \times \mathcal{A}|$ be the number of state action pairs and set $K = \subset$

$\mathbb{R}^d$ to be the convex set of occupancy measures described by Lemma 5.2. Finally, let $\beta > 0$ be such that there exists some exploration policy π_{exp} with $\mu(x, a, \pi_{\text{exp}}) > \beta$ for all states x and actions a.

Input: Step size $\eta > 0$, exploration constant $\delta \in (0, 1]$, approximation constant $c > 0$

Choose $\mu_1 \in K_{\delta\beta}$ arbitrarily;

for *Each episode index* $\tau = 1, \ldots, T$ **do**

  Execute one episode following policy π_t, obtained from μ_t according to Lemma 5.1;

  Estimate the complete reward function $\hat{r}_t$ as in (5.5);

  Set $\mu_{\tau+1/2}(x, a) = \mu_\tau(x, a) \exp(\eta \hat{r}_\tau(x, a))$ for each state x and action a;

  Set $\mu_{\tau+1} = P_{K_{\delta\beta}}(\mu_{\tau+1/2})$, where $P_{K_{\delta\beta}}$ is a black-box that computes c-approximate projections onto $K_{\delta\beta}$ wrt $\|\cdot\|_1$.

end

Algorithm 10: Approximate Online Mirror Ascent for Loop Free Episodic MDPCRs Under Evaluative Feedback

Theorem 5.15. *Let M be a loop free episodic MDPCR, $\pi \in \Pi$ be any Markov policy, and $\pi_1, \ldots, \pi_T$ be the sequence of policies produced by Algorithm 10 with parameters $\delta \in (0, 1]$, $c = \frac{\beta\delta\eta}{\sqrt{T}}$, and $\eta = \sqrt{\frac{D_{\max}}{LT}}$ where L is the number of layers in the MDPCR and $D_{\max} \geq \sup_{\mu \in K_{\delta\beta}} D_J(\mu, \mu(\pi_1))$. Then, the regret of an agent that follows the sequence of policies $\pi_{1:T}$ relative to the fixed policy π is bounded as follows*

$$\sum_{\tau=1}^{T} \left(\mathbb{E}_\pi \left[\sum_{t=1}^{L} r_\tau(X_t, A_t) \right] - \mathbb{E}_{\pi_\tau} \left[\sum_{t=1}^{L} r_\tau(X_t, A_t) \right] \right) \leq 2\sqrt{dTD_{\max}} + \sqrt{T} + L\delta T,$$

and the per-time-step computational cost is $O(H(\beta\delta, c) + d)$, where $H(\beta\delta, c)$ is the cost of the black-box approximate projection routine and d is the number of state-action pairs.

Proof. The proposed algorithm for the evaluative feedback setting is identi-

cal to the instructive feedback setting, with the exception that we estimate the reward function. As in the proof of Theorem 5.14, for any $\mu \in K$, let $\mu_\delta = (1-\delta)\mu + \delta\mu(\pi_{\exp})$ be the mixture of μ with $\mu(\pi_{\exp})$. Then we have

$$\begin{aligned}\sum_{\tau=1}^{T} \hat{r}_\tau^\top(\mu - \mu(\pi_\tau)) &= \sum_{\tau=1}^{T} \hat{r}_\tau^\top(\mu - \mu_\delta) + \sum_{\tau=1}^{T} \hat{r}_\tau^\top(\mu_\delta - \mu(\pi_\tau)) \\ &\le \frac{D_{\max}}{\eta} + \sqrt{T} + L\delta T + \sum_{\tau=1}^{T} \hat{r}_t(\mu(\pi_{t+1/2}) - \mu(\pi_t)).\end{aligned}$$

In the previous proof we bounded the expressions $\hat{r}_t(\mu(\pi_{t+1/2}) - \mu(\pi_t)) \le \eta \|\hat{r}_t\|_\infty^2$ using Lemma 5.12. That is a bad idea, in this case, since the components of $\hat{r}_t$ scale inversely with $\mu(\pi_t)$ (because of the importance weights) and may be very large. Instead, we upper bound the expectation of this term in the following way.

The following lemma is extracted from [AHR2008].

Lemma 5.16. *Let $\mathcal{F}_t$ be a σ-field, w_t and $\hat{r}_t$ be random d-dimensional vectors that are measurable with respect to $\mathcal{F}_t$, and set*

$$w_{t+1/2}(x,a) = w_t(x,a)\exp(\eta\hat{r}_t(x,a))$$

and suppose $\mathbb{E}[\hat{r}_t \,|\, \mathcal{F}_t] = r_t$. Then

$$\mathbb{E}\big[\hat{r}_t^\top(w_{t+1/2} - w_t) \,|\, \mathcal{F}_t\big] \le \eta\mathbb{E}\bigg[\sum_{x,a} w_t(x,a)\hat{r}_t(x,a)^2 \,|\, \mathcal{F}_t\bigg].$$

Now, let $\mathcal{F}_\tau$ denote the sigma algebra generated by the first $\tau-1$ episodes. For each state x and action a, we have

$$\begin{aligned}\mathbb{E}\big[\hat{r}_t(x,a) \,|\, \mathcal{F}_\tau\big] &= \mathbb{E}\big[\mathbb{I}\{X_\ell = x, A_\ell = a\}\,\frac{r_t(x,a)}{\mu(x,a,\pi_\tau)} \,|\, F_\tau\big] \\ &= \frac{r_t(x,a)}{\mu(x,a,\pi_\tau)}\mathbb{E}[\mathbb{I}\{X_\ell = x, A_\ell = a\} \,|\, \mathcal{F}_\tau] \\ &= r_t(x,a).\end{aligned}$$

Therefore, we can apply Lemma 5.16 to get

$$\mathbb{E}\left[\hat{r}_t^\top(\mu(\pi_{t+1/2}) - \mu(\pi_t))\right] \leq \eta d.$$

Substituting this above gives the bound

$$\sum_{\tau=1}^{T} \mathbb{E}[\hat{r}_\tau^\top(\mu - \mu(\pi_\tau))] \leq \eta T d + \frac{D_{\max}}{\eta} + \sqrt{T} + L\delta T.$$

Setting $\eta = \sqrt{\frac{D_{\max}}{Td}}$ gives the optimal bound of

$$\sum_{\tau=1}^{T} \mathbb{E}[\hat{r}_\tau^\top(\mu - \mu(\pi_\tau))] \leq 2\sqrt{TdD_{\max}} + \sqrt{T} + L\delta T.$$

□

As far as the dependence on τ is concerned, by choosing $\delta = 1/\sqrt{T}$, we can thus improve the previous state-of-the-art bound of Neu *et al.* [NGSA2014] that scales as $O(\tau^{3/2}\sqrt{T\ln(|\mathcal{A}|)})$ to $O(\sqrt{\tau T \ln|\mathcal{A}|})$. The update cost of the algorithm of Neu *et al.* [NGSA2014] is $O(|\mathcal{X}|^3 + |\mathcal{X}|^2|\mathcal{A}|)$, while here the cost of our algorithm is $\tilde{O}(T^{1/4}d^{3.5}/\sqrt{\beta})$.

5.3.3 Uniformly Ergodic MDPCRs with Instructive Feedback

Finally, this section introduces an algorithm for learning in uniformly ergodic MDPCRs under instructive feedback. For the rest of this section, fix a uniformly ergodic MDPCR $M = (\mathcal{X}, \mathcal{A}, x_{\text{start}}, \mathcal{P}, (r_t)_{t\in\mathbb{N}})$ with mixing time $\tau < \infty$ and $r_t(x,a) \in [0,1]$ for all states, actions, and times. Let $d = |\mathcal{X}\times\mathcal{A}|$ be the number of state action pairs and set $K \subset \mathbb{R}^d$ to be the convex set of stationary distributions described by Lemma 5.6. Finally, let $\beta > 0$ be such that there exists an exploration policy π_{exp} with $\mu(x,a,\pi_{\text{exp}}) \geq \beta$ for all states and actions.

Theorem 5.17. *Let M be a uniformly ergodic MDPCR with mixing time $\tau < \infty$ and let $\pi \in \Pi$ be any Markov policy and $\pi_1, \ldots, \pi_T$ be the sequence of*

Input: Step size $\eta > 0$, exploration constant $\delta \in (0, 1]$, approximation constant $c > 0$

Choose $\mu_1 \in K_{\delta\beta}$ arbitrarily;
for *Each time index* $t = 1, \ldots, T$ **do**
  Receive state X_t from environment;
  Sample action A_t from π_t, obtained from μ_t according to Lemma 5.5;
  Receive complete reward function r_t from environment;
  Set $\mu_{t+1/2}(x, a) = \mu_t(x, a) \exp(\eta r_t(x, a)$ for each state x and action a;
  Set $\mu_{t+1} = P_{K_{\delta\beta}}(\mu_{t+1/2})$, where $P_{K_{\delta\beta}}$ is a black-box that computes c-approximate projections onto $K_{\delta\beta}$ wrt $\|\cdot\|_1$.
end

Algorithm 11: Approximate Online Mirror Ascent for Uniformly Ergodic Episodic MDPCRs Under Instructive Feedback

policies produced by Algorithm 11 with parameters $\delta \in (0, 1]$, $\eta = \sqrt{\frac{D_{\max}}{T(2\tau+3)}}$, *and* $c = \frac{\beta\delta\eta}{\sqrt{T}}$. *Then the regret of the agent that follows policies* π_t *at time* t *relative to policy* π *can be bounded as*

$$\mathbb{E}_\pi\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] - \mathbb{E}_{\pi_{1:T}}\left[\sum_{t=1}^{T} r_t(X_t, A_t)\right] \leq 2\sqrt{(2\tau+3)TD_{\max}} + \sqrt{T} + \delta T + 4\tau + 4.$$

Proof. This proof is essentially identical to the proof of Theorem 5.14 and has been omitted. □

According to Bubeck *et al.* [BCK2012], for online bandit linear optimization over a compact action set $K \subset \mathbb{R}^d$, it is possible to obtain a regret of order $O(d\sqrt{T \log T})$ regardless of the shape of the decision set K, which, in our case would translate into a regret bound of order $O(|\mathcal{X} \times \mathcal{A}|\sqrt{T \log T})$. Whether the algorithm proposed in this paper can be implemented efficiently depends, however, on the particular properties of K: Designing the exploration distribution needed by this algorithm requires the computation of

the minimum volume ellipsoid containing K and this problem is in general NP-hard even when considering a constant factor approximation [N2007].

Selecting $\pi_{\exp}(x,\cdot)$ to be the uniform distribution, $\beta > 0$ and $D_{\max} \le L\ln(|\mathcal{A}|)$, results in a $O(\sqrt{dLT\ln(|\mathcal{A}|)})$ bound on the regret for $\delta = 1/\sqrt{T}$, while the time-complexity of the algorithm is still $O(d^{3.5}T^{1/4}/\sqrt{\beta})$ as in the full-information case. Neu *et al.* considered the same problem under the assumption that any policy π visits any state with probability at least α for some $\alpha > 0$, that is, $\inf_\pi \sum_a \mu(x,a) \ge \alpha > 0$. They provide an algorithm with $O(d)$ per round complexity whose regret is $O(L^2\sqrt{T|\mathcal{A}|\ln(|\mathcal{A}|)/\alpha})$. Compared to their result, we managed to lift the assumption $\alpha > 0$, and also improved the dependence on the size of the MDP, while paying a price in terms of increased computational complexity.

Chapter 6

Conclusion

This thesis documents the two projects that I worked on during my MSc program. Both projects contribute to the goal of building computer systems that are capable of learning for themselves to solve problems and succeed at tasks. Each project focuses on specific mathematical questions related to a formal learning problem.

The first project addresses the question of which baseline function to use in policy gradient reinforcement learning methods for Markov decision processes. The baseline function's role is to alter the performance gradient estimate used internally by policy gradient methods. I show that if the formal learning objective is a concave function of the agent's policy parameters, then the regret of a policy gradient method can be upper bounded by a quantity that only depends on the baseline function only through the second moment of the gradient estimates. This suggests that the baseline function should be chosen to minimize the second moment of the gradient estimates, which I show to be equivalent to the more intuitive notion of minimizing the mean squared error of the gradient estimates. I derive closed form expressions for this baseline in terms of the MDP transition probability kernel, the agent's policy, and the agent's policy parameterization. Since the MDP transition probability kernel is unknown to the agent, I also propose two algorithms for estimating this baseline while interacting with the environment. Finally, I present a preliminary empirical comparison

of the always-zero baseline, the value function baseline, and my proposed baseline. This comparison demonstrates a statistically significant increase in performance when using my proposed baseline, as long as we are careful to initialize our estimates reasonably accurately.

The goal of the second project is to design new learning algorithms for MDPCRs. The main difference between MDPCRs and standard MDPs is that, in the former, the environment chooses the sequence of reward functions in an adversarial manner. This difference makes it easier to model some real-world problems as MDPCRs, especially those with non-stationary dynamics. I propose three new algorithms, all based on an approximate version of online mirror ascent: one for learning in loop-free MDPCRs under instructive feedback, one for learning in loop-free MDPCRs under evaluative feedback, and one for learning in uniformly ergodic MDPCRs under instructive feedback. Each of these algorithms has regret bounds that either improve or complement the regret bounds of existing algorithms, and which often hold even under weaker assumptions on the environment. In the development of these algorithms, it was necessary to analyze an approximate version of online mirror ascent, where the projection step is only computed approximately. To my knowledge, this is the first rigorous analysis of this approximation to online mirror ascent, despite the fact that the projection step can often only be approximated.

Both projects provide sound, theoretically justified answers to important questions in the fields of reinforcement learning and online learning.

Bibliography

[AHR2008] Abernethy, J., Hazan, E., & Rakhlin, A. "Competing in the Dark: An efficient algorithm for bandit linear optimization" in *Proceedings of the 21st Annual Conference on Learning Theory* (July 2008): 263–274.

[BCK2012] Bubeck, S., Cesa-Bianchi, N., & Kakade, S. M. "Towards minimax policies for online linear optimization with bandit feedback" in *Journal of Machine Learning Research - Proceedings Track*, (2012): 23:41.1–41.14

[DGS2014] Dick, T., György, A., & Szepesvári, Cs., "Online learning in Markov Decision Processes with Changing Cost Sequences" in *Proceedings of The 31st International Conference on Machine Learning* (2014): 512–520.

[EKM2005] Even-Dar, E., Kakade, S. M., & Mansour, Y. "Experts in a Markov Decision Process" in *Advances in neural information processing systems*, Vol. 17 (2005): 401–408.

[EKM2009] Even-Dar, E., Kakade, S. M., & Mansour, Y. "Online markov decision processes" in *Mathematics of Operations Research*, Vol. 34, No. 3 (2009): 726–736.

[GBB2004] Greensmith, E., Bartlett, P., & Baxter, J. "Variance reduction techniques for gradient estimates in reinforcement learning" in *The Journal of Machine Learning Research* Vol. 5 (2004): 1471–1530.

[GPS2014] György, A., Pál, D., & Szepesvári, Cs. "Online learning: Algorithms for big data" (2014): `https://www.dropbox.com/s/bd38n4cuyxslh1e/online-learning-book.pdf`.

[N2007] Nemirovski, A. "Advances in convex optimization: Conic programming" in *Proceedings of International Congress of Mathematicians*, Vol. 1 (2007): 413–444.

[NGSA2014] Neu, G., György, A., Szepesvári, Cs., & Antos, A. "Online markov decision processes under bandit feedback" in *IEEE Transactions on Automatic Control*, Vol. 59, No. 3 (February 2014): 676–691.

[NGS2010] Neu, G., György, A., & Szepesvári, Cs. "The online loop-free stochastic shortest-path problem" in *Proceedings of the 23rd Annual Conference on Learning Theory* (June 2010): 231–243.

[S2012] Shalev-Shwartz, S. "Online learning and online convex optimization" in *Foundations and Trends in Machine Learning*, Vol. 4, No. 2 (2012): 107–194.

[SB1998] Sutton, R. S. & Barto, A. G. *Reinforcement learning: An Introduction*. Cambridge, MA: The MIT Press, 1998.

[SMSM2000] Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. "Policy gradient methods for reinforcement learning with function approximation" in *Advances in Neural Information Processing Systems*, Vol. 12. Cambridge, MA: The MIT Press, 2000: 1057–1063.

[Cs2010] Szepesvári, Cs. *Algorithms for Reinforcement Learning* [Synthesis Lectures on Artificial Intelligence and Machine Learning 9]. San Rafael, CA: Morgan & Claypool Publishers, 2010.

[W1992] Williams, R. J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning" in *Machine Learning*, Vol. 8, No. 3-4 (1992): 229–256.

[YMS2009] Yu, J. Y., Mannor, S., & Shimkin, N. "Markov decision processes with arbitrary reward processes" in *Mathematics of Operations Research*, Vol. 34, No. 3 (2009): 737–757.