Linear Programming in Reinforcement Learning

Patrick Nadeem Ward

Computer Science
McGill University, Montreal

April 13, 2021

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Science.

Abstract

Recent advances in Reinforcement Learning (RL) have been primarily focused on leveraging the field of Deep Learning (DL) to overcome issues when scaling to larger sequential decision-making problems. An alternative approach which has historically been used to understand certain aspects of RL is the Linear Programming (LP) formulation of this problem. The LP approach to RL provides a unique perspective and strong theoretical guarantees. This thesis attempts to provide a comprehensive review of the use of LPs in RL and to demonstrate how they have helped lay the foundations for creating broader connections between RL and convex optimization. This thesis also proposes a kernel-based LP approach applicable to large, and even infinite state spaces.

Résumé

Les progrès récents en matière d'apprentissage par renforcement (RL) ont été principalement axés sur l'exploitation des outils de l'apprentissage profond (DL) afin de surmonter les problèmes liés à l'élargissement des domaines dans les problèmes de prise de décision séquentielle. Malgré cela, des travaux antérieurs qui utilisent la programmation linéaire (LP) sont actuellement poursuivi en raison de leurs perspectives uniques et leurs solides garanties théoriques. Cette thèse tente de fournir un examen complet de l'utilisation de la programmation linéaire dans le cadre de l'apprentissage par renforcement. Il s'agit d'une demonstration de comment cette approche a fourni des bases pour la création de connexions entre l'apprentissage par renforcement et l'optimisation convexe. Cette thèse propose également une approche en programmation linéaire base sur les méthodes de noyaux en apprentissage par renforcement, applicable aux grands espaces d'état.

Acknowledgements

To my advisor, Doina Precup: I'm extremely thankful to have been given the opportunity of a lifetime to pursue my passion. You have always encouraged my ideas and taught me how to turn them into valid research directions. I take great pride and consider myself very fortunate to have been advised by you. Your constant positivity, support and guidance have helped me grow tremendously and for this I'll be forever grateful. You are an inspiration!

To my mentor, Vincent Francois-Lavet: Your patience and diligence with my early efforts were fundamental in developing my capacity to do research. You nurtured my curiosity and taught me how to ask the right questions.

To my friends and colleague: I consider myself fortunate to be surrounded by so many incredibly intelligent and caring individuals. I've learnt so much from them and their importance cannot be overstated. To Aly Ibrahim, for being my wingman throughout my masters and always pushing me to do better. To Joey Bose, for teaching me how to do research on a day-by-day level and for being an outlet for all of my stupid thoughts and ideas. To Ariella Smofsky, for being an incredibly kind and hardworking co-author. To Riashat Islam, for sharing my passion and giving me confidence in my ideas. To Maxime Wabartha, for challenging my thought and reminding me how to be rigorous.

To my mother, Nadwa Rafeh, and my brother, Ziad Ward: Your undying love has been an endless source of strength that has helped me persevere through any and all challenges in my life. I would be lost without you. You are my role models and my home.

Finally, to Joelle Azouri: For making these some of the best years of my life. You give my life happiness and balance.

Contribution of Authors

- All of the research conducted and writing of the chapters in this thesis was my own work. This work was carried out with input and advice by my advisor, Doina Precup.

Contents

1

Introduction

The field of Reinforcement Learning (RL) has garnered considerable attention in recent years. Several recent works by Minh et al. [41], Silver et al. [62], Schrittwieser et al. [60], OpenAI [49] and Abbeel et al. [2] have demonstrated the validity of RL as a framework for solving large scale sequential decision-making problems. However, these successes aren't without their limitations. Some well-known limitations involve data inefficiency [32], the lack of theoretical guarantees when using function approximation [28], [48], as well as off-policy divergence [5]. These issues can manifest themselves, for example, as premature convergence to sub-optimal solutions, extremely resource intensive algorithms or, in the context of safety in RL, as taking actions deemed dangerous. Avoiding dangerous actions is crucial in the robotics settings where they may lead to the loss of valuable equipment or even life. To remedy some of these issues, researchers have drawn on different fields in hopes of finding connections that can be usefully applied. One such attempt is the Linear Programming (LP) approach to RL which formulates the Sequential Decision-Making problem as an LP.

There are several reasons why the use of LP in RL is important. First, it provides a unique perspective for solving sequential decision-making problems, which differs from classical RL [64]. For example, the interpretation of the dual variables as an *occupancy measure* of an agent in different states of the environment may help

uncover new approaches such as the work on Successor Representations [18]. Second, it can be used as a starting point from which to create connections between the vast, well-established, theories and techniques of convex optimization (for which LP is a sub-field) and RL which may hopefully be leveraged to advance the field of RL. Thirdly, the LP approach may provide further justification to concepts already present within the RL literature as demonstrated by Neu et al. [47] with a framework that proves powerful performance guarantees for a state-of-the-art algorithm, TRPO [61]. Additionally, the LP approach is well suited to the study of Constrained Markov Decision Processes [3] for its simplicity in adding constraints.

This thesis aims to review and synthesize the literature on LPs and their use in advancing the field of RL. More specifically, the objectives of this thesis are:

(i) Review the literature to study the applicability of LPs to the sequential decision-making problem and compile the existing theory, methods and techniques.

(ii) Demonstrate that we may derive a kernel-based LP based on the concept of kernel-based RL(Ormoneit and Sen [50]) which simultaneously alleviates the reliance on knowledge of the environment and dependence on the size of the state space when scaling to larger, even infinite, state spaces.

Over the years, the use of LPs in RL went through several important conceptual developments. Linear Programming was pioneered in the mid 1900s by Dantzig, Von Neumann and Tucker among others [17]. Around the same time, the development of the larger theory surrounding convex optimization was studied by Rockafeller [57]. These advancement lead to the first use of Linear Programming in Reinforcement Learning in the 1960-70s with the work of Manne [39], De Ghellinck [26] and Denardo [23]. This work was later revived by Puterman [53] and developed for the linear function approximation case by De Farias and Van Roy [21,

20]. Using LP duality, Wang et al. [72] demonstrated how to re-derive equivalent methods to those commonly used in Reinforcement Learning (such as value iteration and temporal-differencing) based solely on the dual domain. Pazis and Parr [51] suggested a modification to the approximate Linear Program [21] that replaces the linear approximation with a non-parametric representation. More recently, Chen and Wang [16] and Bas-Serrano and Neu [8] leveraged the subsequent Lagrangian derived from the Linear Programming formulation to demonstrate provably efficient stochastic gradient descent policy optimization. The line of work of Nachum et al. [44, 43, 42] leveraged duality and convex regularization for the Linear Programming formulation to try to obtain new algorithms and convergence guarantees in the batch data, off-policy setting.

This thesis is organized in six chapters. Chapter 2 provides an overview of the sequential decision-making problem and convex optimization as the underlying frameworks for RL and LP. This lays the groundwork for the subsequent chapters. Chapter 3 provides a literature review of LPs for Markov Decision Processes. This chapter will be divided into two sections: Section 3.1 discusses the formulation of the LP for solving the "planning" problem in RL (with access to the environment dynamics). Section 3.2 describes extensions used to scale the LP approach to larger, more complex problems. Chapter 4 introduces a kernel-based LP which extends the LP approach to the RL setting with a possibly infinite number of states. Chapter 5 summarizes recent attempts to use the LP formulation as a starting point from which to create new connections to the optimization literature that can be leveraged to obtain new algorithms, insights and theoretical guarantees. Chapter 6 concludes and offers recommendations for future directions.

2

Background

This chapter provides an overview of the fundamentals of RL as a class of methods which aim to handle sequential decision making under uncertainty (section 2.1). Furthermore, this chapter also provides background on convex optimization (section 2.2), which underpins the theory useful for understanding Linear Programming.

The experienced reader may use this chapter as a reminder while simultaneously getting acquainted with the notation. The new reader may use this chapter as a guide to key concepts in these fields as well as a reference to aid with further study.

2.1 Sequential Decision Making under Uncertainty

Sequential Decision Making is an integral part of artificial intelligence concerned with specifying the behavior of an agent interacting with a possibly uncertain environment over an extended period of time. According to Littman [37], *"Sequential decision making [...] is the act of answering the question "What should I do now?" when "now" is one of a finite set of states, "do" is one of a finite set of actions, "should" is maximize a long-run measure of reward, and "I" is an automated planning or learning system (agent)."*

One of the main frameworks used to study this concept rigorously is the *Markov Decision Process* (MDP), first introduced by Bellman [11].

2.1.1 Markov Decision Process

In an MDP, an *agent* interacts with an *environment* at a sequence of discrete times t by taking actions $A_t \in \mathcal{A}$ while being in states of the environment $S_t \in \mathcal{S}$. In doing so, the agent receives a reward $R_{t+1} = r(S_t, A_t)$ and transitions to another state S_{t+1}. The process is then repeated. The components of this interaction can be exemplified through Figure 2.1[1].

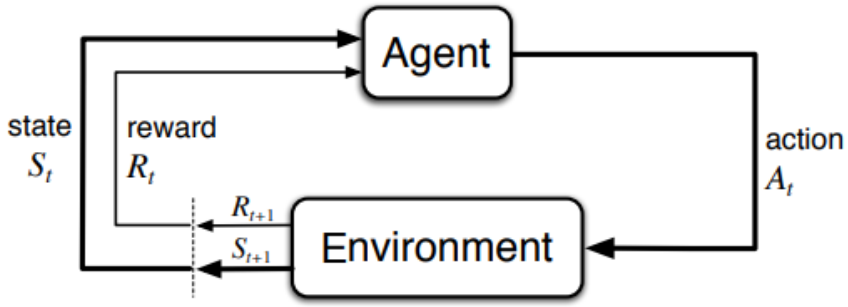

Figure 2.1: The agent-environment interaction in a Markov Decision Process.

Definition 1. *A **Markov decision process** (MDP), denoted $\mathcal{M}$, is a 5-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ where $\mathcal{S}$ is the set of states (often called the state space), $\mathcal{A}$ the set of actions (often called the action space), $\mathcal{P}$ the set of transition probability distributions $P(.|s, a)$ defined for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$, r is the reward function typically defined over state-action pairs $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $\gamma \in (0, 1)$ the discount factor.*

An MDP *is* a Markov process, hence the **Markov property** is assumed to hold, which means that transitions to new states, conditioned on the current state, are independent of the preceeding sequence of states:

$$P(S_{t+1}|S_0, A_0, S_1, A_1, \ldots, S_t, A_t) = P(S_{t+1}|S_t, A_t).$$

The goal of an agent in this setting is to produce a **policy** π which establishes how it should act. A policy is therefore just an instruction manual on what to do in

[1]Figure taken from Sutton and Barto [64].

each state and can either be a **deterministic** mapping, $\pi : \mathcal{S} \to \mathcal{A}$, or **stochastic**, by defining a probability distribution over the set of actions conditioned on a state $\pi(a|s) = P(A_t = a|S_t = s)$.

As a result of its broad definition, many problems can be formulated and solved using the MDP framework presented above. The following are a few popular illustrative examples:

- A mobile autonomous robot cleaner makes the decision of whether to continue searching for more trash to collect or return to its recharging station. [59]
- An inventory management problem for determining the optimal reorder points of a product. A store must determine how much of a product to order (the action) based on how much stock is left (the state) in order to maximize the long-run profit while mitigating inventory carrying costs. [53]
- A baseball pitcher makes the decision of what type of pitch to throw. Depending on the opponents, the pitcher would like to throw the most likely pitch that leads to strikeout while reducing the risk of a home run, in order to optimize the global goal of maximizing the chances of winning the game. [37]

When specifying an MDP modelling a specific application, several dimensions of the environment have to be considered: finite versus infinite state and action spaces, discounted ($\gamma \in (0, 1)$) versus undiscounted ($\gamma = 1$) objective, and finite vs infinite horizon. Moving forward, this thesis will adopt the following assumptions on MDPs unless stated otherwise.

Assumption 1. *The reward function is bounded, i.e.*

$$\max_{(s,a)} |r(s, a)| = R_{\max} < \infty.$$

Assumption 2. *The sets of states and actions are considered discrete, finite and fixed over time, $\mathcal{S}_t = \mathcal{S}$ and $\mathcal{A}_t = \mathcal{A}$ for all time steps t.*

Assumption 3. *The MDP is assumed to have an infinite horizon and discounted returns, ie* $\gamma \in (0, 1)$.

Assumption 4. *The policies are assumed to be* ***stationary****, i.e. they are invariant over time,* $\pi(A_t = a|S_t = s) = \pi(a|s)$ *for all time steps* t.

Assumption 5. *The MDP is assumed to be* ***ergodic****. Informally, this means that any state is reachable from any other state with some positive probability, regardless of the policy. More specifically, this means that the transitions induced by any policy will produce an ergodic Markov Chain (i.e. irreducible and positive recurrent).*

2.1.1.1 Value Function

The notion of a value function in RL is ubiquitous. To make decisions, an agent must be able to compare different outcomes. The value function allows for this comparison by quantifying how valuable a particular state (or state-action pair) is. As such, a value function is a mapping from state (or state-action) to a real valued scalar, representing the expected sum of per-step rewards received from this point onward, ie. the **return** $\mathcal{R}$:

$$\mathcal{R}_t = \sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \tag{2.1}$$

However, the rewards received are dependent on the states visited and the actions taken. Hence the expected return is dependent on a specific policy as well as the environment, both of which are possibly stochastic. This leads to the notion of a value function. More formally,

Definition 2. *The* ***state-value function*** *of a (stationary) policy* π *is the conditional expected return given a current state* s.

$$v_\pi(s) = \mathbb{E}_\pi[\mathcal{R}_0|S_0 = s] = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s \right], \; s \in \mathcal{S} \tag{2.2}$$

A slight modification to this definition leads to another notion of interest, the action-value function, which differs only in that it imposes the first action a to be taken.

Definition 3. *The **action-value function** of a (stationary) policy π is the conditional expected return given an initial state-action pair:*

$$q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s, A_0 = a \right], \; s \in \mathcal{S}, a \in \mathcal{A} \tag{2.3}$$

While the previous two quantities look similar, further elaboration in the subsequent chapters will clarify the purpose of this distinction.

2.1.1.2 Policy Evaluation

To evaluate the quality of a policy, its value function must be computed. One important initial observation when unrolling the sum of rewards in (2.2) is that we can write the value function recursively.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s \right] \\ &= \mathbb{E}\left[r(S_0, A_0) + \sum_{t=1}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s \right] \\ &= \mathbb{E}\left[r(S_0, A_0) + \sum_{t=0}^{\infty} \gamma^{t+1} r(S_{t+1}, A_{t+1}) \,\middle|\, S_0 = s \right] \\ &= \mathbb{E}\left[r(S_0, A_0) + \gamma \sum_{t=0}^{\infty} \gamma^t r(S_{t+1}, A_{t+1}) \,\middle|\, S_0 = s \right] \\ &= \mathbb{E}_\pi \left[r(S_0, A_0) + \gamma v_\pi(S_1) \mid S_0 = s \right], \end{aligned} \tag{2.4}$$

We can therefore write the state-value function recursively as:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_\pi(s') \right) \tag{2.5}$$

Similarly, unrolling the action-value function leads to a recursive formula:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = x, A_0 = a \right] \\ &= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_\pi(s') \end{aligned} \tag{2.6}$$

These two equations (2.4) and (2.6) are commonly referred to in the literature as *Bellman equations*. Other terms have been given to these equations, such as *policy evaluation equations* [22], however this thesis will use the rather pedantic terminology of ***Bellman recurrence equations*** to clearly distinguish them from the Bellman optimality equations which will be introduced shorty.

Writing these equations in matrix form leads to a straightforward solution to policy evaluation. Considering the state-value function and reward as vectors in $\mathbb{R}^{|\mathcal{S}|}$, the linear systems of equations can be written in a more condensed matrix form as:

$$v^\pi = r^\pi + \gamma P^\pi v^\pi, \tag{2.7}$$

here $r^\pi(s) = \sum_a \pi(a|s) r(s,a)$ and $P^\pi(s,s') = \sum_a \pi(a|s) P(s'|s,a)$. Notice that fixing a policy for an MDP induces a Markov chain described by the $\mathcal{S} \times \mathcal{S}$ stochastic matrix P^π. We can therefore solve for v^π in the above equation.

$$\begin{aligned} & v^\pi = r^\pi + \gamma P^\pi v^\pi \\ \Leftrightarrow\ & (I - \gamma P^\pi) v^\pi = r^\pi \\ \Leftrightarrow\ & v^\pi = (I - \gamma P^\pi)^{-1} r^\pi \end{aligned} \tag{2.8}$$

Hence the existence of the value function relies on the invertibility of the matrix $(I - \gamma P^\pi)$. The interested reader can find a formal proof of its existence in Bacon's thesis [4]. However, this approach is usually avoided in practice since the computational complexity for inverting a matrix is of the order $\mathcal{O}(|\mathcal{S}|^3)$ operations, a limiting factor when the state space becomes very large.

Alternatively, an indirect iterative approach can be taken that leads to successively better approximation. This approach consists of defining a linear operator $\mathcal{T}^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ (usually called the one-step Bellman operator) as follows:

$$\mathcal{T}^\pi v = r^\pi + \gamma P^\pi v \tag{2.9}$$

The vector v that satisfies $\mathcal{T}^\pi v = v$ is our v^π of interest. Therefore, determining the value function is equivalent to finding the fixed point of the operator $\mathcal{T}^\pi$. It can be shown that successive application of the operator $\mathcal{T}^\pi$ on an initially random vector converges to a (unique) fixed-point [4]. This may be established by demonstrating that the operator is a contraction mapping and using Banach fixed-point theorem [6]. The complete proof is outside the scope of this thesis, however, it can be found in Bacon [4] and Bertsekas and Tsitsiklis [12]. Algorithm 1 describes the resulting algorithm.

Algorithm 1: Iterative Policy Evaluation

Input: MDP $\mathcal{M}$, π and γ ;

Compute r^π, where $r^\pi(s) = \sum_a \pi(a|s) r(s,a)$;

Compute P^π, where $P^\pi(s,s') = \sum_a \pi(a|s) P(s'|s,a)$;

initialize $v = \mathbf{0} \in \mathbb{R}^{|\mathcal{S}|}$;

repeat

  $v \leftarrow r^\pi + \gamma P^\pi v$;

until *convergence criteria*;

2.1.1.3 Policy Improvement / Control

The previous section demonstrated how to evaluate the significance of a given policy. However, our ultimate objective is to improve the policy in order to reach an ***optimal policy*** with the highest expected value (sometimes called the *control* problem in the literature).

Definition 4 (Optimal Policy)**.** *A policy π^* is **optimal** when the value function associated to this policy is greater than the value function associated to any other policy. i.e. if $\pi \neq \pi^*$ then, $v^{\pi^*}(s) \geq v^\pi(s)\ \forall s \in \mathcal{S}$.*

The optimal value function may be written recursively. In addition to adhering to the Bellman recurrence equations (2.7), the optimal value function must

also satisfy a nonlinear system of equations:

$$v^* = \max_{\pi} \{r^\pi + \gamma P^\pi v^*\} \tag{2.10}$$

It can be shown that there exists at least one optimal stationary policy that is deterministic [53]. Given that such a deterministic policy exists, we may write the above equation in component form as:

$$v^*(s) = \max_{a \in \mathcal{A}} q^*(s,a) = \max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v^*(s') \right\} \tag{2.11}$$

And similarly, for the optimal action value function:

$$q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a' \in \mathcal{A}} q^*(s',a') \tag{2.12}$$

These two equations (2.11) and (2.12) are known as the ***Bellman optimality equations***. Retrieving the optimal policy once the optimal state-value (or action-value) function has been computed is relatively straightforward:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v^*(s') \right) \tag{2.13}$$

which is a *greedy* policy based on v^*.

An iterative method for finding the optimal policy can be derived using the Bellman optimality equations. Following the same steps as in Section 2.1.1.2, we can define the nonlinear Bellman optimality operator $\mathcal{T}^* : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ as:

$$\mathcal{T}^* v = \max_{\pi} \{r^\pi + \gamma P^\pi v\} \tag{2.14}$$

Based on equation (2.10), the fixed point of this operator, $\mathcal{T}^* v^* = v^*$, satisfies the Bellman optimality equations and is therefore the optimal value function of interest. Similar to our discussion on the convergence of Iterative Policy Evaluation (Algorithm 1), we can prove that the successive application of this operator on an initially random vector converges to its fixed point, by establishing that $\mathcal{T}^*$ is a contraction mapping. The proof is left outside of this thesis but can be found in Bellman's seminal paper [11]. This leads to the *value iteration algorithm*.

Algorithm 2: Value Iteration

Input: an MDP $\mathcal{M}$;

Output: estimated optimal action-value function q

initialize $q = \mathbf{0} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$;

repeat

| $q(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a' \in \mathcal{A}} q(s',a'), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$

until *change in one iteration is small*;

The update equation used as the premise for the value iteration algorithm (2.11) can be divided into two separate steps, leading to a new algorithm, as follows:

$$v^*(s) = r(s, \pi^*(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi^*(s)) v^*(s') \tag{2.15}$$

$$\text{where } \pi^*(s) = \arg\max_{a \in \mathcal{A}} (r(s,a) + \gamma P(s'|s,a) v^*(s')) \tag{2.16}$$

These two equations can be seen as a step of *policy evaluation* (2.15) followed by a step of *policy improvement* (2.16). This realization leads to the methods of **policy iteration** [27] where policy evaluation and policy improvement are repeatedly applied in order to get monotonically improving policies and value functions (this result is known as the *policy improvement theorem* [64]). Value iteration can be seen as a special case of policy iteration where we approximately compute v^* using only a one-step Bellman update. In general, we may decide to do m iterations of policy evaluation before proceeding to policy improvement (2.16). This method is known as **modified policy iteration** (MPI).

2.1.2 Using Reinforcement Learning to solve MDPs

So far, we have solved MDPs under the assumption of complete knowledge of the environment, known as the *"planning"* scenario (also called the *Dynamic Programming* (DP) setting). This assumption is limiting in practice because, for many environments, this information is hard to acquire or model accurately. The *Reinforcement*

Learning (RL) setting bypasses this limitation by learning strictly through experience (trial-and-error). In doing so, the agent must decide whether to gain new information about the unknown environment (explore) or use its current information about the environment to gain rewards (exploit). This problem is commonly referred to as the **exploration versus exploitation trade-off**. The notion of trial-and-error learning is unique to RL and distinguishes this field from others such as Supervised Learning and Unsupervised Learning. The following sections present Monte-Carlo (MC) and Temporal Difference (TD) learning methods for evaluation and control in the RL scenario.

2.1.2.1 Monte Carlo Methods

Broadly speaking, MC methods are concerned with using random samples to obtain numerical results [56]. In our setting, MC methods will be used to estimate the state/action-value function. If the MDP is episodic, then we can estimate $v_\pi(s)$ (the expected return) as the average return received over multiple sampled episodes starting at state s. As the number of samples tends towards infinity, the strong law of large numbers tells us that the average return is an unbiased estimator for the mean.

$$v_\pi(s) = \mathbb{E}_\pi[\mathcal{R}_i | S_0 = s] \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{R}_i, \text{ (for large } n) \tag{2.17}$$

where $\mathcal{R}_i$ are sampled returns from episodes following state s. We may proceed in a similar fashion to estimate $q_\pi(s, a)$ by taking the average return of sampled episodes starting in state-action pair (s, a).

$$q_\pi(s, a) = \mathbb{E}_\pi[\mathcal{R}_0 | S_0 = s, A_0 = a] \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{R}_i, \text{ (for large } n) \tag{2.18}$$

Once we obtain values for $q_\pi(s, a)$ we may proceed to improve the policy using the generalized policy iteration principle (seen in section 2.1.1.3) to get successively better policies. However, this assumes that we are able to sample from *any* initial state-action pair (called exploratory starts), which isn't always applicable in practice. For

example, many environments, such as typical games, may only be initiated at a designated starting state. Another way of maintaining exploration is through ϵ-*greedy* policies that take random actions with some small probability, ϵ, rather than strictly greedy action selection. In doing so, we guarantee that all state-action pairs will be sampled enough (in the limit) to make proper estimates of the values functions over all states.

For efficient implementations of MC methods, an incremental update formula for our estimate can be devised. Let $\hat{v}_n(s)$ be our estimate for $v_\pi(s)$ after n samples, this can be rewritten incrementally as:

$$\begin{aligned}
\hat{v}_n(s) &= \frac{1}{n}\sum_{i=1}^{n}\mathcal{R}_i \\
&= \frac{1}{n}\mathcal{R}_n + \frac{1}{n}\sum_{i=1}^{n-1}\mathcal{R}_i \\
&= \frac{1}{n}\mathcal{R}_n + \frac{n-1}{n}\hat{v}_{n-1}(s) \\
&= \frac{1}{n}\left(\mathcal{R}_n + (n-1)\hat{v}_{n-1}(s)\right) \\
&= \hat{v}_{n-1}(s) + \frac{1}{n}(\mathcal{R}_n - \hat{v}_{n-1}(s))
\end{aligned} \tag{2.19}$$

Given a new sample, we can now update the estimate in $\mathcal{O}(1)$ instead of $\mathcal{O}(n)$. Moreover, we may interpret the fraction $\frac{1}{n}$ as the amount of weight we give to the most recent sample. This parameter is referred to as the *step-size* (or *learning rate*) and can be altered more generally to any α_n. A well-known result in stochastic approximation theory gives us conditions required to guarantee convergence with probability 1 [55].

$$\sum_{n=1}^{\infty}\alpha_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty}\alpha_n^2 < \infty \tag{2.20}$$

A decaying step size, such as $1/n$, will satisfy these conditions. However, a constant $\alpha \in (0, 1]$ does not. Nonetheless, a constant step-size is commonly used in practice and is actually desirable in non-stationary environments.

The form of the update rule (2.19) occurs so frequently that it's components are given

names. The general form, as specified in [64], is

$$NewEstimate \leftarrow OldEstimate + StepSize\Big[Target - OldEstimate\Big] \tag{2.21}$$

The term between brackets, $[Target - OldEstimate]$, is referred to as the *error* in the estimate.

2.1.2.2 Temporal Difference

Temporal Difference (TD) methods combine the idea of sampling from experience (section 2.1.2.1) and updating based on previous estimates (section 2.1.1.3) to form a particularly useful approach for solving MDPs in the RL setting, first introduced by Sutton [63].

Just as the MC estimate (equation 2.19) can be seen as an approximation of the expected value (2.2), $v_\pi(s) = \mathbb{E}[\mathcal{R}|S_0 = s]$, so can the TD estimate (equation 2.22) be seen as an approximation of the expectation of the value function written in the form of the *Bellman recurrence equations* (2.4), $v_\pi(s) = \mathbb{E}_\pi[r(s_0, a_0) + \gamma v_\pi(S_1) \mid S_0 = s]$. Written as an incremental update similar to (2.21), The target value used for a TD estimate replaces a sampled return from an episode, $\mathcal{R}_i$, (as used in MC estimate, equation 2.19) with a sampled one-step reward.

$$\hat{v}(S_t) \leftarrow \hat{v}(S_t) + \alpha\left(R_{t+1} + \gamma\hat{v}(S_{t+1}) - \hat{v}(S_t)\right) \tag{2.22}$$

S_t and R_{t+1} are capitalized to emphasis the fact that these are random variables produced through one sampled transition. The estimate produced by this method is called *TD(0)* (or *one-step TD*) and is classified as a *bootstrap estimate* since it reuses previous estimates to compute new ones. The error term $R_{t+1} + \gamma\hat{v}(S_{t+1}) - \hat{v}(S_t)$ is typically called the *TD-error*

The TD(0) update rule (2.22) gives us a way to estimate v_π that provides several benefits over MC methods. First, this method can be executed

in an online fashion where we update our estimate as soon as we get a new reward. This is different from MC methods that require the entire episode to terminate before updating. In doing so, TD(0) can learn value functions quickly when MDPs have very long, even infinite, horizons. Second, computation is distributed in a much more even manner when compare to MC methods since simple updates happen at every step of an MDP rather than waiting until the end of an episode to process all of the new information. Finally, information is propagated faster using TD(0) since learning happens *within* an episode, and therefore new information can instantly be used when updating the value function. The convergence of TD(0) in the tabular setting was first proven by Sutton [63] and later strengthened by Dayan [19] and Jaakkola et al. [29].

We can extend this approach beyond one-step reward estimates. One might question why we should limit ourselves to consider only a one-step reward estimate. Why not use a 2 or 3-step estimate? This idea leads to *n-step TD* methods [73] which use an n-step reward as the target value in our update equation (2.21). Let $G_{t:t+n}$ be the multi-step target written as:

$$G_{t:t+n} = \sum_{i=0}^{n-1} \gamma^i R_{t+i+1} + \gamma^n v(S_{t+n}) \tag{2.23}$$

Where the subscript for the return G, $G_{t:t+n}$, describes the range of time-steps for which sampled rewards are used before applying bootstrapping. When $n = 1$, this term is equivalent to the TD(0) target. Similarly, when $n = T$, where T is the length of the episode, this becomes the MC target. Although MC updates are unbiased, they lead to high variance due to the possibly very long and stochastic trajectories. TD(0) target estimates, on the other hand, produce low variance estimates that have a high bias due to bootstrapping. Therefore, n-step TD methods unify these two ideas and allow us to explicitly regulate the bias-variance trade-off between them.

Taking this a step further, we may decide not to truncate our reward signal (the

target) at some fixed number n. Instead, we may want to average different n-step targets. *TD(λ)* [73] can be understood as creating an exponentially weighted moving average over n-step updates, with λ being the decay rate. The target used in this method is called the *λ-return*,

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n} \tag{2.24}$$

Setting $\lambda = 0$ recovers TD(0) while setting $\lambda = 1$ reduces to the MC target.

Having discussed policy evaluation using TD methods, we now focus on TD policy improvement. The methods discussed are akin to the previously seen sections. Under the general process of policy iteration, we may use updates similar to (2.22) to obtain estimates for $q_\pi(s,a)$ and proceed to improve our ϵ-greedy policy using these new estimates. This method is known as *Sarsa* [58] (n-step Sarsa when using n-step target updates). *Q-learning* [73], which can be seen as a sample-based approach to value iteration (previously seen as Algorithm 2), directly approximates the optimal action-value function q^* through the update equation,

$$q(S_t,A_t) \leftarrow q(S_t,A_t) + \alpha\left(R_{t+1} + \gamma\max_{a'} q(S_{t+1},a') - q(S_t,A_t)\right), \tag{2.25}$$

The resulting algorithm yields:

Algorithm 3: Q-learning

Input: step size $\alpha \in]0, 1]$, small ϵ;

Output: An optimal action-value function q^*;

initialize $q(s, a)$ arbitrarily $\forall s \; \forall a$;

for *each episode* **do**

initialize s;

for *each step of episode* **do**

Choose action a from state s using policy derived from q;

Take action a, observe reward r and next state s'

$q(s, a) \leftarrow q(s, a) + \alpha \left(r + \gamma \max_{a'} q(s', a') - q(s, a)\right)$;

$s \leftarrow s'$

end

end

Interestingly, the q values are updated independently of the policy being followed. Therefore, *any* policy can be followed so long as we guarantee that all state-action pairs continue to be visited. This is known as *off-policy* learning and differs from the *on-policy* learning of Sarsa or MC methods which directly evaluate the policy being followed.

2.2 Convex Optimization

> *"Everyone goes through a stage of intellectual development where they realize that **everything** is an optimization problem. [...] My recommendation is to get over this phase quickly. What matters is **what type** of optimization problem it is, because basically most optimization problems you can't solve."*
>
> *– Stephen Boyd, Lecture 1 - Convex Optimization I (EE 364A), Stanford, 2009*

A mathematical optimization problem is formalized as maximizing or minimizing

an objective function subject to constraints. *Convex optimization* problems are a special class of optimization problems for which there exists comprehensive theory [57] and can be solved numerically very efficiently [46]. This section is derived from Boyd and Vandenberghe [14] and follows similar notation.

2.2.1 Terminology and Theory of Convex Optimization

We describe a mathematical optimization problem using the following notation:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{2.26}$$

With $x \in \mathbb{R}^n$ the *optimization variables*, $f_0(x) : \mathbb{R}^n \to \mathbb{R}$, the *objective function* (also called the cost function), $f_i(x) \leq 0,\ i = 1, \ldots, m$ the *inequality constraints* and $h_i(x) = 0,\ i = 1, \ldots, p$ the *equality constraints*. Problem (2.26) may also have no constraints in which case it's called an *unconstrained* optimization problem. The form of the optimization problem (2.26), which consists of minimizing an objective subject to inequality constraints of the form, $f_i(x) \leq 0,\ i = 1, \ldots, m$, and equality constraints of the form, $h_i(x) = 0,\ i = 1, \ldots, p$, is known as a *standard form* optimization problem in the literature. Many optimization problems can be manipulated and re-expressed in a similar form to (2.26). For example, a maximization problem may be turned into a minimization problem by negating the objective function. Additionally, we may also use techniques such as a change of variables or introducing new constraints in order to transform our optimization problem into the standard form. Therefore, we focus our attention only on standard form problems (2.26).

To better understand the characteristics needed for the above problem to be considered convex, the following key concepts are defined:

Definition 5 (Convex sets)**.** *A set C is **convex** if the line segment between any two points in C also lies in C. i.e. if $x_1, x_2 \in C$, and $\theta \in [0, 1]$ then,*

$$\theta x_1 + (1 - \theta)x_2 \in C \tag{2.27}$$

Intuitively, this can be understood as saying that the line segment joining any two points in the set must also belong to the set (Figure 2.2).

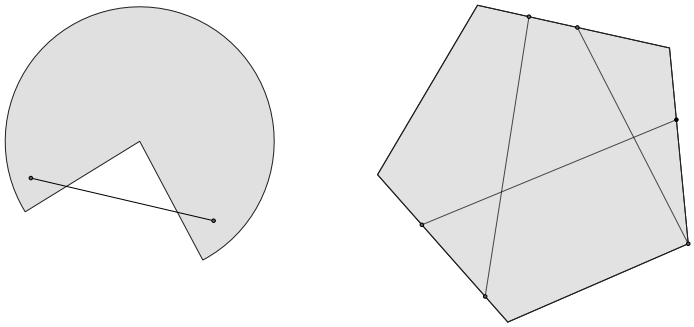

Figure 2.2: Convex and nonconvex sets. *Left.* the semicircle is not convex since the line segment between the two points in not contained in the set. *Right.* The polygon is a convex set since all line segments connecting two points are contained in the set.

A point of the form $\theta_1 x_1 + \cdots + \theta_n x_n$, with $0 \leq \theta_i \leq 1 \ \forall i$ and $\sum_{i=1}^{n} \theta_i = 1$, is called a ***convex combination***. It can be shown that a set is convex if and only if it contains every convex combination of its points [14].

Definition 6 (Convex function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if for all $x_1, x_2 \in$ **dom**(f) and $\theta \in [0, 1]$ we have*

$$f(\theta x_1 + (1 - \theta)x_2) \leq \theta f(x_1) + (1 - \theta)f(x_2). \tag{2.28}$$

Intuitively, this means that the line segment joining two points of the function must be larger than the function between those two points (Figure 2.3).

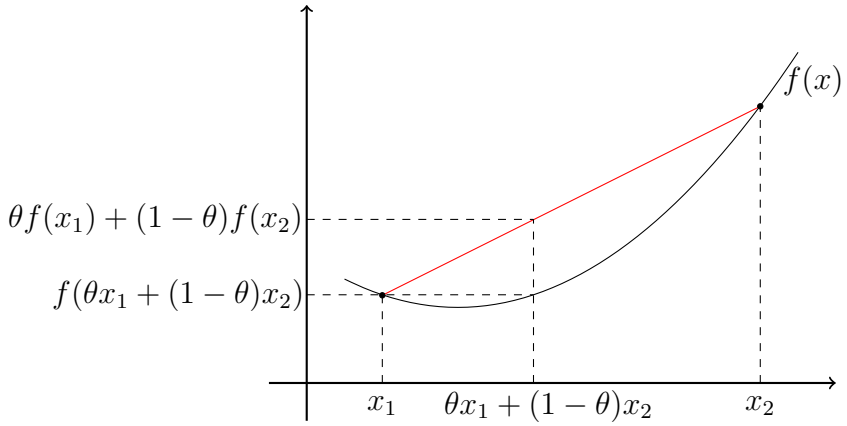

Figure 2.3: Plot of a convex function.

Convex functions have properties which are conducive to finding global optima. Having quickly defined the main building blocks for the theory of convex optimization, we now consider some of their unique properties.

Definition 7. *(First order conditions for convexity) Assuming f is differentiable, then f is convex if and only if* ***dom****(f) is convex and*

$$f(x_2) \geq f(x_1) + \nabla f(x_1)^T (x_2 - x_1) \quad \forall x_1, x_2 \in \textbf{\textit{dom}}(f) \tag{2.29}$$

Where $\mathbf{dom}(f)$ is the domain set of the function. This condition implies that the tangent plane at point x_1, $y = f(x_1) + \nabla f(x_1)^T (x_2 - x_1)$, is a global under-estimator of the function f (Figure 2.4). Knowing only local information about a point (it's value and it's derivative), we can make global statement about the function. This is a key realization about convex functions that leads to the well-known condition for optimality; if f is convex and $\nabla f(x_1) = 0$ then $\forall x_2 \in \mathbf{dom}(f)$, $f(x_2) \geq f(x_1)$, meaning that x_1 is the global minimum of the function f.

Definition 8. *(Second-order conditions for convexity)*

Assume f is twice differentiable, then f is convex if and only if ***dom****(f) is convex and the Hessian of f is positive semidefinite, i.e,*

$$\nabla^2 f(x) \succeq 0 \quad \forall x \in \textbf{\textit{dom}}(f) \tag{2.30}$$

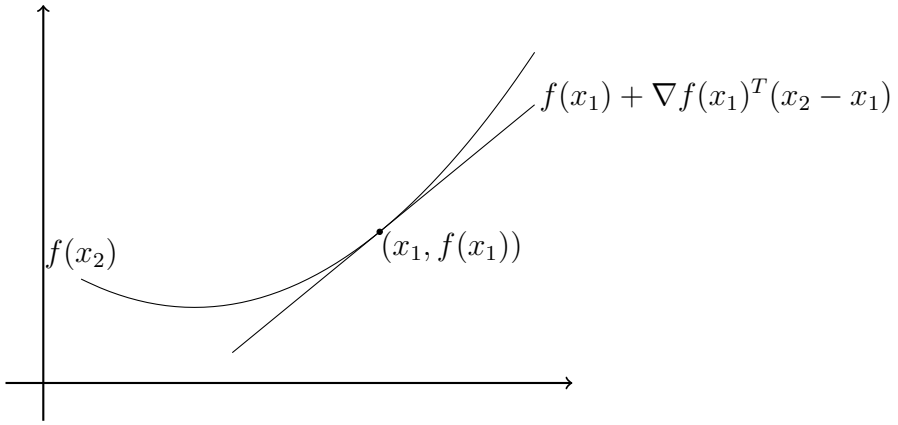

Figure 2.4: Visualization of the first order condition for convexity.

A matrix H is positive semidefinite if and only if $x^T Hx \geq 0,\ \forall x \in \mathbb{R}^n$. Intuitively, this can be understood as the operator H applied to the vector x keeps it the same direction as x originally (the dot product between the two vectors $\langle x, Hx \rangle \geq 0$). The second order condition for convexity is often expressed as saying the function f has positive curvature. We will introduce a slight abuse of notation, $\succeq$ when applied to matrices is as described above whereas, when applied to vectors refers to element-wise greater than or equal to.

We may now define a *convex optimization* problem as the optimization problem (2.26), when the objective function is convex, the inequality constraints functions, $f_1, \ldots, f_m$, are convex and the equality constraints are affine. Namely,

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & a_i^T x = b_i, \quad i = 1, \ldots, p, \end{aligned} \tag{2.31}$$

Where $f_0, \ldots, f_m$ are convex functions. When the objective function and constraints are all affine, then the problem is called a **Linear Program** (LP) and has the general

form:

$$\begin{aligned} \text{minimize} \quad & c^T x + d \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b, \end{aligned} \tag{2.32}$$

Geometrically, the linear program can be interpreted as pushing a ball within the feasible region as far as possible in the direction of the vector $-c$ (Figure 2.5).

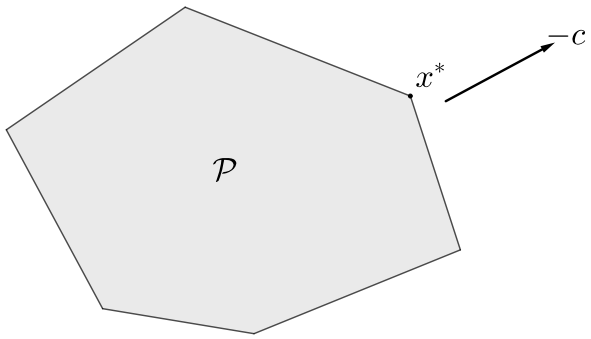

Figure 2.5: Geometric interpretation of an LP. x^* is the point that goes furthest in the direction $-c$ within the feasible region $\mathcal{P}$.

The first practical algorithm for solving LPs was the simplex method conceived by Dantzig [17]. However, it was later shown to have worst-case exponential time complexity [34]. Khachiyan [36] and Karmarkar [33] later developed algorithms which improved the time complexity to polynomial in the worst case. Nesterov and Nemirovski extended these results to nonlinear convex optimization [46] thus demonstrating efficient solution methods to convex optimization problems.

2.2.2 Duality

Going back to the optimization problem considered (2.26). The ***Lagrangian*** for this problem augments the objective function by incorporating the constraints into one equation.

Definition 9 (Lagrangian)**.** *The* ***Lagrangian*** *associated with problem (2.26),* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$*, is defined as*

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \tag{2.33}$$

Derived from the method of Lagrange multipliers for optimization problems with equality constraints, we refer to the variables λ and ν as the *Lagrange multipliers* or *dual variables*. Each dual variable is associated with one constraint. This equation (2.33) has meaningful properties which can be used to help find optimal values (most famous of which are the KKT conditions).

Definition 10 (Lagrange dual function)**.** *Sometimes just called the dual function,* $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$*, is defined as the minimum value of the Lagrangian with respect to* x*,*

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) = \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right) \tag{2.34}$$

The dual function (2.34) has a few interesting properties. First, note that the dual function (2.34) is just the infimum of an assortment of affine functions with respect to λ and ν. Since the dual function is the infimum of affine functions, therefore it is *always* concave (even when the problem (2.26) is not convex). Second, for any $\lambda \succeq 0$ and any ν, the dual function gives a lower bound on the optimal value to the original problem (2.26). To see this, suppose that $\tilde{x}$ is a feasible point for problem (2.26), i.e. $f_i(\tilde{x}) \leq 0$ and $h_i(\tilde{x}) = 0$, then we have

$$\sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{x}) \leq 0$$

Accordingly, it follows that,

$$L(\tilde{x}, \lambda, \nu) = f_0(\tilde{x}) + \sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{x}) \leq f_0(\tilde{x})$$

which indicates that the infimum is definitely smaller, i.e. $g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) \leq f_0(\tilde{x})$. This is true for any arbitrary $\tilde{x}$ therefore it's also true for the optimal point

p^*, $g(\lambda, \nu) \leq p^*$. The dual function gives a *non-trivial* lower bound on the optimal solution to (2.26).

Based on the above, one can inquire about the best possible lower bound. This leads to another optimization problem:

$$\begin{aligned} \underset{\lambda,\nu}{\text{maximize}} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \succeq 0. \end{aligned} \tag{2.35}$$

This is called the ***Lagrange dual problem*** (or simply the dual problem for short) associated with the ***primal problem*** (2.26). This problem is always concave and provides a useful lower bound on the primal problem (2.26).

Example 1. *Consider trying to find the dual for an LP of the form:*

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \textit{subject to} \quad & Ax \preceq b. \end{aligned} \tag{2.36}$$

The Lagrangian is $L(x, \lambda) = c^T x + \lambda^T (Ax - b) = -b^T \lambda + (c + A^T \lambda)^T x$ *with* $\lambda \succeq 0$. *The dual function is therefore,*

$$g(\lambda) = \inf_x L(x, \lambda) = -b^T \lambda + \inf_x (c + A^T \lambda)^T x$$

Notice that $g(\lambda)$ *is* $-\infty$ *almost always. The exception being when* $c + A^T \lambda = 0$. *The (Lagrange) dual problem of the LP (2.36) can therefore be reformulated explicitly as*

$$\begin{aligned} \text{minimize} \quad & -b^T \lambda \\ \text{subject to} \quad & A^T \lambda = -c \\ & \lambda \succeq 0, \end{aligned} \tag{2.37}$$

which is itself another LP. Moreover, it can be shown that taking the dual of this new LP leads to the original LP (2.36). Lagrangian duality is described in much more detail in the work of Rockafellar [57].

In general, we would like to know the *optimally gap* between the primal problem (2.26) and its corresponding dual problem (2.35), i.e. $p^* - d^*$ with p^* and d^* the primal and

dual optimal values respectively. ***Weak duality*** states that $d^* \leq p^*$ regardless of whether the problem is convex or not. ***Strong duality*** refers to when the primal and dual optimal values are equal, $p^* = d^*$ [71]. Fortunately, for most convex optimization problems strong duality holds (conditions for which strong duality holds are called *constraint qualification conditions*, with *Slater's condition* being one such example). Furthermore, it can be shown that strong duality holds for *any* LP provided that either the primal or dual problem are feasible [14].

Strong duality can be used to derive the condition of ***complementary slackness***. Suppose strong duality holds and let x^* and (λ^*, ν^*) be primal and dual optimal points respectively. This implies that

$$
\begin{aligned}
f_0(x^*) &= g(\lambda^*, \nu^*) && (2.38)\\
&= \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i^* f_i(x) + \sum_{i=1}^{p} \nu_i^* h_i(x) \right) && (2.39)\\
&\leq f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* f_i(x^*) + \sum_{i=1}^{p} \nu_i^* h_i(x^*) && (2.40)\\
&\leq f_0(x^*) && (2.41)
\end{aligned}
$$

The first line (2.38) follows from strong duality, the second line (2.39) is by definition of the dual function and the third (2.40) is true because the dual function is the infimum (therefore it will be at least as small as x^*). The last inequality follows from the fact that x^* is a feasible point meaning that $f_i(x^*) \leq 0$ and $h_i(x^*) = 0$. Notice that all above equations in fact hold with strict equality. Therefore, we *must* have

$$\sum_{i=1}^{m} \lambda_i^* f_i(x^*) = 0.$$

since each term in this sum is nonpositive, we conclude that $\lambda_i^* f_i(x^*) = 0$ for each i. This condition is called complementary slackness and we can deduce from it the following properties.

$$\text{if } \lambda_i^* > 0 \Rightarrow f_i(x^*) = 0; \quad \text{if } f_i(x^*) < 0 \Rightarrow \lambda^* = 0 \tag{2.42}$$

Strong duality can be interpreted as the *saddle-point* of the Lagrangian function (2.33). Consider a convex optimization problem without equality constraints, the optimal value of the primal problem can be expressed as,

$$p^* = \inf_x \sup_{\lambda \succeq 0} L(x, \lambda) \tag{2.43}$$

Since $\sup_{\lambda \succeq 0} L(x, \lambda) = f_0(x)$ whenever x is primal feasible. By definition of duality, we also have the optimal value of the dual problem,

$$d^* = \sup_{\lambda \succeq 0} \inf_x L(x, \lambda) \tag{2.44}$$

Therefore, strong duality implies,

$$\sup_{\lambda \succeq 0} \inf_x L(x, \lambda) = \inf_x \sup_{\lambda \succeq 0} L(x, \lambda). \tag{2.45}$$

A saddle-point for a function $f : W \times Z \to \mathbb{R}$ (with $W \subseteq \mathbb{R}^n$ and $Z \subseteq \mathbb{R}^n$) is a point such that,

$$f(\tilde{w}, z) \leq f(\tilde{w}, \tilde{z}) \leq f(w, \tilde{z}), \quad \forall w \in W \; z \in Z, \tag{2.46}$$

meaning that $\tilde{w}$ minimizes $f(w, \tilde{z})$ on the right hand side and $\tilde{z}$ maximizes $f(\tilde{w}, z)$ on the left hand side. If the function is convex-concave in w and z, this implies that strong duality hold (2.45) at $f(\tilde{w}, \tilde{z})$. We can therefore deduce that if the pair (x, λ) is a saddle-point of the Lagrangian, then they are primal and dual optimal and the optimality gap is zero.

3

Linear Programming in Reinforcement Learning

This chapter provides a review of direct approaches for using LPs in RL. By examining the original implementation as well as highlighting a selected number of modern extensions, we provide a clear and comprehensive picture of the current state of this area of research. Until now, Chapter 2 provided an overview of two separate fields of mathematics, namely Sequential Decision-Making (section 2.1) and Convex Optimization (section 2.2). This chapter demonstrates how these two fields may be linked together to the benefit of RL.

We begin by introducing the formulation of an LP for solving MDPs and its corresponding dual LP (section 3.1), which provide a unique perspective and strong theoretical guarantees. Afterwards, we illustrate several distinct extensions that attempt to broaden the applicability of this approach while maintaining some of the favorable theory of LPs (section 3.2).

3.1 The Linear Programming formulation for solving MDPs

The original formulation of an LP for solving MDPs was derived by Manne [39] and Denardo [23] in the undiscounted infinite horizon case, and by De Ghellinck [26] in the discounted case. Puterman [53] elaborated further on this idea and provided a

concise explanation of this approach summarized below.

The LP formulation for solving MDPs can be deduced through properties of the value function. Recall the Bellman optimality operator (2.14), $\mathcal{T}^* v = \max_\pi \{r^\pi + \gamma P^\pi v\}$, which can be interpreted as nonlinear constraints on the optimal state-value function v^*. Interstingly, we may find an upper bound, $v \geq \mathcal{T}^* v$, by considering a larger set of linear constraints.

$$v(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a) v(s'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S} \tag{3.1}$$

To see this, consider the above linear inequalities (3.1) with respect to just one state s,

$$\left.\begin{array}{l} v(s) \geq r(s,a_1) + \gamma \sum_{s'} P(s'|s,a_1) v(s') \\ v(s) \geq r(s,a_2) + \gamma \sum_{s'} P(s'|s,a_2) v(s') \\ \vdots \\ v(s) \geq r(s,a_{|\mathcal{A}|}) + \gamma \sum_{s'} P(s'|s,a_{|\mathcal{A}|}) v(s') \end{array}\right\} |\mathcal{A}| \text{ equations for fixed } s \in \mathcal{S}. \tag{3.2}$$

Notice that the right hand side of the above equation, $r(s,a) + \gamma \sum_{s'} P(s'|s,a) v(s')$, is nothing more than the one-step Bellman recurrence equation (2.4) using a deterministic policy $\pi(a|s) = 1$. The set of linear inequalities (3.2) state that the value function $v(s)$ is larger than the one-step Bellman recurrence equation using *any* deterministic policy. Therefore, the value function is also larger than any convex combination of the above Bellman recurrence equations, i.e.

$$v(s) \geq \sum_{\forall a} \pi(a|s) \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) v(s') \right) \tag{3.3}$$

Since this is true for any arbitrary policy π therefore it is also true for the optimal policy π^*. However, the optimal value function v^* adheres to strict equality for the optimal Bellman operator, $v^* = \mathcal{T}^* v^*$. Already knowing that such a solution is guaranteed to exist [53], therefore we're interested in the minimum vector upper

bound v such that $v \geq \mathcal{T}^* v$. This leads to the following optimization problem,

$$\begin{aligned} \underset{v \in \mathbb{R}^{|\mathcal{S}|}}{\text{minimize}} \quad & \sum_{\forall s} \mu(s) v(s) \\ \text{subject to} \quad & v(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a) v(s'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S} \\ & v(s) \text{ unconstrained}, \forall s \in \mathcal{S} \end{aligned} \tag{3.4}$$

Where μ can be interpreted as the starting/initial state distribution. Notice that the objective and constraints for (3.4) are linear, therefore the optimization problem (3.4) is an LP with $|\mathcal{S}|$ variables and $|\mathcal{S}| \times |\mathcal{A}|$ constraints. We may write it in a more compact matrix form as:

$$\begin{aligned} \underset{v \in \mathbb{R}^{|\mathcal{S}|}}{\text{minimize}} \quad & \mu^T v \\ \text{subject to} \quad & v \geq r^a + \gamma P^a v, \quad \forall a \in \mathcal{A} \\ & v \text{ unconstrained} \end{aligned} \tag{3.5}$$

With $r^a \in \mathbb{R}^{|\mathcal{S}|} : [r(s_1,a).r(s_2,a), \ldots, r(s_{|\mathcal{S}|},a)]^T$ and $P^a \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|} : P^a_{i,j} = P(j|i,a)$ the rewards and transitions associated with action a. We will call this LP (3.4) the LP-MDP. Following Example 1 (from section 2.2), we may write the dual LP (after some algebraic manipulation) for the LP-MDP (3.4) as,

$$\begin{aligned} \underset{d}{\text{maximize}} \quad & \sum_{\forall s \in \mathcal{S}} \sum_{\forall a \in \mathcal{A}} r(s,a) d(s,a) = \mathbb{E}_{(s,a) \sim d}[r(s,a)] \\ \text{subject to} \quad & \sum_{\forall a \in \mathcal{A}} d(s,a) - \gamma \sum_{\forall s' \in \mathcal{S}} \sum_{\forall a \in \mathcal{A}} P(s|s',a) d(s',a) = \mu(s), \quad \forall s \in \mathcal{S} \\ & d(s,a) \geq 0, \ \forall a \in \mathcal{A}, \forall s \in \mathcal{S} \end{aligned} \tag{3.6}$$

The dual problem (3.6) has $|\mathcal{S}|$ constraints and $|\mathcal{S}| \times |\mathcal{A}|$ variables, d. We will call this the Dual LP-MDP.

The Dual LP-MDP (3.6) provides a unique perspective for solving MDPs. Theorem (6.9.1) from Puterman [53] establishes an alternative way to express the dual constraints. Assuming d is any dual feasible solution and using the

following policy,

$$\pi(a|s) = \frac{d(s,a)}{\sum_{\forall a'} d(s,a')} \tag{3.7}$$

Then, $d(s,a)$ is equivalent to:

$$d(s,a) = \sum_j \mu(j) \sum_{t=0}^{\infty} \gamma^t P(S_{t+1} = s, A_{t+1} = a | S_0 = j)$$

Also, if we let $u(s) = \sum_{\forall a \in \mathcal{A}} d(s,a)$, we may write the constraints for (3.6) as:

$$\begin{aligned}
\mu(s) &= u(s) - \gamma \sum_{\forall s'} \sum_{\forall a} p(s|s',a) d(s',a) \\
&= u(s) - \gamma \sum_{\forall s'} \sum_{\forall a} p(s|s',a) d(s',a) \frac{u(s')}{\sum_{\forall a'} d(s',a')} \\
&= u(s) - \gamma \sum_{\forall s'} \sum_{\forall a} p(s|s',a) \pi(a|s') u(s') \\
&= u(s) - \gamma \sum_{\forall s'} \sum_{\forall a} P^{\pi}(s|s') u(s')
\end{aligned} \tag{3.8}$$

which in matrix form may be written as $\mu^T = u^T \Big(I - \gamma P_{\pi} \Big)$. Solving for u^T, we get

$$u^T = \mu^T \Big(I - \gamma P^{\pi} \Big)^{-1} = \mu^T \Big[\sum_{t=0}^{\infty} (\gamma P^{\pi})^t \Big]$$

The term $\sum_{t=0}^{\infty} (\gamma P^{\pi})^t$ forms a valid *Neumann series* since the spectral radius of γP, $\rho(\gamma P)$, is strictly less than 1. Therefore, the matrix $(I - \gamma P^{\pi})^{-1}$ exists and $(I - \gamma P^{\pi})^{-1} = \sum_{t=0}^{\infty} (\gamma P^{\pi})^t$. This is more thoroughly derived as part of corollary C.4 from Puterman [53]. Writing both of these results again,

$$d^{\pi}(s,a) = \sum_j \mu(j) \sum_{t=0}^{\infty} \gamma^t P^{\pi}(S_t = s, A_t = a | S_0 = j) = \mathbb{E}_{\mu} \left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\{S_t = s, A_t = a\}} \right] \tag{3.9}$$

$$u^{\pi}(s) = \sum_j \mu(j) \sum_{t=0}^{\infty} \gamma^t P^{\pi}(S_t = s | S_0 = j) = \mathbb{E}_{\mu} \left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\{S_t = s\}} \right] \tag{3.10}$$

These quantities, called the discounted state visitation functions, represents a sort of *occupancy measure* for the expected amount of time the system will occupy state s (or both s and a) under the initial state distribution μ. Since these quantities are the dual variables for the associated LP-MDP, they provide an alternative way to solve

MDPs which differs from the value function based approaches seen previously (section 2.1). It's worth noting that they are not distributions:

$$\sum_{s'}\sum_{t=0}^{\infty}\gamma^t P^\pi(S_t = s'|s) = \sum_{t=0}^{\infty}\gamma^t \sum_{s'} P^\pi(S_t = s'|s) = \frac{1}{1-\gamma}.$$

Some may prefer to work with distributions and therefore sometimes consider a normalized variant such as $\bar{u}^\pi = (1-\gamma)u^\pi$ which is referred to as the *discounted state distribution* [66]. In the infinite horizon undiscounted case, the occupancy measure (3.10) becomes a *stationary distribution* for the underlying Markov Chain induced by a policy (assuming all stationary policies lead to irreducible Markov Chains).

The discounted state visitation functions (3.9) and (3.10) share many of the same properties as the value function and state-action value function. They may be written recursively (equation (3.8)) and can be solved explicitly using the methods seen in chapter 2 such as temporal difference learning. This approach was developed by Dayan [18] and is called *Successor Representation* (SR) for its explicit emphasis on comparing states based on how often they encounter similar states in the future, their successors. This representation is said to improve generalization to different tasks for a given environment [18]. The SR perspective has been subject to recent interest in the field of RL with the works of Barreto [7], Kulkarni [35].

There are two ways to derive the optimal policy from the dual LP-MDP (3.6). First, we can solve for the optimal dual variables and derive a policy based on normalizing the dual variables per state, similar to equation (3.7). Second, by strong duality the optimal value of both the primal LP-MDP (3.4) and dual LP-MDP (3.6) are equal: $\sum_{\forall s\in\mathcal{S}}\mu(s)v^*(s) = \sum_{\forall s\in\mathcal{S}}\sum_{\forall a\in\mathcal{A}} r(s,a)d^*(s,a)$. By complimentary slackness, we can also say the following about the properties of these optimal variables.

$$d^*(s,a)\left(v^*(s) - \gamma\sum_{s'}P(s|s,a)v^*(s') - r(s,a)\right) = 0, \quad \forall a \in A,\ \forall s \in S$$

Recall that by definition of the stationary deterministic optimal policy we have

$$v^*(s) - \gamma \sum_{s'} P(s'|s, \pi^*(s))v^*(s') - r(s, \pi^*(s)) = 0$$

Since π^* is deterministic, we also have that for any other action a,

$$v^*(s) \geq \gamma \sum_{s'} P(s'|s, a)v^*(s') - r(s, a) \quad \forall a \neq \pi^*(s)$$

Therefore, we must have that $d^*(s, a) = 0$ for all $a \neq \pi^*(s)$ by complementary slackness. Thus, the optimal dual variable, d, only has $|\mathcal{S}|$ nonzero elements (one per state) corresponding to the $|\mathcal{S}|$ active row constraints. This suggests a correspondence between the optimal policy and the optimal dual variable, $\pi^*(s) = a$ if $d(s, a) > 0$. Finding the optimal policy corresponds to finding the non-zero dual variables.

The LP formulation has theoretical benefits. Since both the primal LP-MDP (3.4) and dual LP-MDP (3.6) are linear programs, they can be solved exactly in polynomial time in the number of variables, constraints and number of bits needed to represent the dynamics of the environment [38].More recently, there have been several positive results which demonstrate that the simplex method [17] (an algorithm for solving LPs), with a specific pivoting rule, is *strongly* polynomial for solving both fixed discount and deterministic MDPs [75] [52]. *Strongly* polynomial in this context refers to the algorithm being polynomial time in only the number of states and actions (and not dependent of the size of the representations of the environment). However, this approach is practical only in small environments. As the state space grows, the dependency on the size of the state space becomes a limiting factor.

3.2 Modern Approaches

As we attempt to tackle larger MDPs, the size of the state space typically grows exponentially. This phenomenon is known as the *curse of dimensionality* [10] and renders most problems intractable. Although the LP approach discussed in section 3.1 has

algorithms with polynomial time theoretical guarantees, the degree of the polynomial is often large enough that the implementation becomes impractical.

Moving forward, this section examines how the LP approach has been extended to be more practical at scale. To do so we study two distinct approaches, namely the Approximate Linear Program [21] (subsection 3.2.1) and the non-parametric Approximate Linear Program [51] (section 3.2.2), which attempts to alleviate some of the issues with scaling while maintaining strong theoretical guarantees on performance.

3.2.1 LP Linear Value Function Approximation

In De Farias and Van Roy [21], the authors approximate the value function by a linear function $\hat{v}(s,\theta) \approx v(s)$, where $\hat{v}(s,\theta)$ is a linear function of the weight vector, θ. In the function approximation setting, for each state s, there is a corresponding *feature vector* $x(s)$, which is a multi-dimensional vector associated with features the agent perceives when in state s. The linear function approximation $\hat{v}(s,\theta)$ is the inner product between these two vectors, $\hat{v}(s,\theta) = \theta^T \cdot x(s)$ with $|\theta| = |x(s)| = n << |\mathcal{S}|$. In matrix form, we can write the value function approximation as a vector $\hat{v} = X\theta$ where X is a $\mathcal{S} \times n$ matrix with each column corresponding to one feature of the feature vector across all states. Notice that the features are in fact *basis functions* for the linear function approximation since they form a linear basis for the set of representable functions using $\hat{v}$. The authors transform the primal LP (3.4) into an approximate LP (ALP) of the from:

$$\begin{aligned} \underset{\theta \in \mathbb{R}^n}{\text{minimize}} \quad & \mu^T \cdot X\theta \\ \text{subject to} \quad & X\theta(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a) X\theta(s'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S} \\ & \theta \text{ unconstrained} \end{aligned} \tag{3.11}$$

Where $X\theta(s)$ is the s^{th} entry of the $\mathcal{S}$-dimensional vector $X\theta$. Notice the number of variables of the ALP (3.11) has been reduced to n.

Having established the ALP, the authors then go on to study the properties and performance guarantees that can be achieved using this approximate solution method. First, they demonstrate the significance of the initial state distribution on the performance of the ALP through the following illuminating Lemma.

Lemma 3.2.1. *(Lemma 1 from De Farias and Van Roy [21])*

A vector $\tilde{\theta}$ *solves*

$$\begin{aligned} \underset{\theta \in \mathbb{R}^n}{\text{minimize}} \quad & \mu^T X\theta \\ \text{subject to} \quad & X\theta \geq r^a + \gamma P^a X\theta, \quad \forall a \in \mathcal{A} \end{aligned} \tag{3.12}$$

if and only if it solves

$$\begin{aligned} \underset{\theta \in \mathbb{R}^n}{\text{minimize}} \quad & \|v^* - X\theta\|_{1,\mu} \\ \text{subject to} \quad & X\theta \geq r^a + \gamma P^a X\theta, \quad \forall a \in \mathcal{A} \end{aligned} \tag{3.13}$$

Where $\|.\|_{1,\mu}$ is the *weighted* ℓ_1 norm (i.e. $\|x\|_{1,c} = \sum_{\forall x_i} |x_i| c_i$). This lemma tells us that our original LP is equivalent to minimizing a weighted norm determined by starting state distribution, μ. The starting state distribution μ can be understood as imposing a trade-off in the quality of the approximation across different states.

The following theorems establish a bound on the quality of this approximation. The first theorem provides a relationship between the solution to the LP (3.11) and the resulting policy derived from this solution. The quality of the approximation is based upon the following evaluation criteria:

$$\mathbb{E}_{s \sim \mu} \left[v^*(s) - v^{\pi}(s) \right] = \|v^* - v^{\pi}\|_{1,\mu}$$

Theorem 3.2.2. *(Theorem 1 from De Farias and Van Roy [21])*

Let $v : S \to \mathbb{R}$ *such that* $v \geq \mathcal{T}^* v$ *(a feasible solution to the LP). Then,*

$$\|v^* - v^{\pi_v}\|_{1,\mu} \leq \frac{1}{1-\gamma} \|v^* - v\|_{1,u^{\pi_v}}$$

In other words, if the approximate value function v found by solving the LP is close to v^* (the right hand side of the above equation) then the performance of the policy generated by v, π_v, should similarly be close to the performance of the optimal policy (the left hand side of the equation). The quantity u^{π_v} refers to the discounted state distribution mentioned in section 3.1 (3.10). Combining this result with the above lemma 3.2.1, we would like the starting state, μ, to be as close to u^{π_v} as possible. The next theorem provides a bound on the error of the ALP.

Theorem 3.2.3. *(Theorem 2 from De Farias and Van Roy [21])*

Let μ be a probability distribution over states. If $\tilde{\theta}$ is an optimal solution to the approximate LP, we have

$$\left\|v^* - X\tilde{\theta}\right\|_{1,\mu} \leq \frac{2}{1-\gamma} \min_{\theta} \|v^* - X\theta\|_\infty$$

This error bound demonstrates the relationship between the quality of the solution to the ALP (left hand side), $X\tilde{\theta}$, and the best possible linear value function approximation (right hand side). If the optimal value function v^* lies close to the span of the basis function defined by x then the result obtained from the ALP will similarly be close. Having said this, we may now bound the performance of the policy derived from the solution to the ALP as,

$$\|v^* - v^{\pi_{X\tilde{\theta}}}\|_{1,\mu} \leq \frac{2}{(1-\gamma)^2} \min_{\theta} \|v^* - X\theta\|_\infty \tag{3.14}$$

Where $\tilde{\theta}$ is the solution to the ALP. This bound assumes $\mu = u^{\pi_{X\tilde{\theta}}}$.

Constraint sampling can be used to reduce the number of constraints in the ALP (3.11). Despite reducing the number of variables in the LP (3.4) to $n << |\mathcal{S}|$ using the linear ALP (3.11), the dependency on $|\mathcal{S}|$ still exists in the number of constraints ($|\mathcal{S}| \times |\mathcal{A}|$ constraints). To alleviate this issue, De Farias and Van Roy [20] consider the effect of *constraint sampling* of the linear ALP (3.11). The idea is to reduce the number of constraints in the ALP by only considering those that belong

to a subset, $\mathcal{X}$, generated by independently sampling m constraints following some distribution over state-action pairs. Additionally, the authors add a parameter $\mathcal{N}$ which restricts the magnitude of the solution. The resulting linear program is named the reduced linear program (RLP) [20],

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \mu^T X\theta \\ \text{subject to} \quad & X\theta(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a) X\theta(s'), \quad \forall (s,a) \in \mathcal{X} \\ & \theta \in \mathcal{N} \end{aligned} \tag{3.15}$$

The authors break down and bound two sources of error for this approach.

The first source of error when solving the RLP (3.15) is the feasibility of the resulting solution. The RLP only adheres to a subset of the constraints, $\mathcal{X}$, whereas the original ALP has all $|\mathcal{S}| \times |\mathcal{A}|$ constraints. To bound this source of error, the authors consider an LP with constraints of the form:

$$\alpha_z^T \theta + k_z \geq 0, \quad \forall z \in Z \tag{3.16}$$

It is easy to see that constraints of this form corresponds directly to the RLP where α_z and k_z would translate to $\alpha_z = (I - \gamma P^a)x$ and $k_z = r^a$ in the RLP (3.15). The following theorem bounds the possibility that a solution using a subset, $\mathcal{X}$, of constraints of the form (3.16) is infeasible with respect to constraints not sampled during the sampling process, $\mathcal{S} \times \mathcal{A} \setminus \mathcal{X}$.

Theorem 3.2.4. *(Sample complexity of near feasibility, Theorem 2.1 from De Farias and Van Roy [20])*

For any $\delta, \epsilon \in]0,1[$ *and*

$$m \geq \frac{4}{\epsilon}\left(n \ln \frac{12}{\epsilon} + \ln \frac{2}{\delta}\right) \tag{3.17}$$

a set $\mathcal{X}$ of m i.i.d. random variables drawn from $|\mathcal{S}| \times |\mathcal{A}|$ according to distribution ψ satisfy

$$\left[\sup_{\{\theta | \alpha_z^T \theta + k_z \geq 0 \ \forall z \in \mathcal{X}\}} \psi(\{y : \alpha_y^T \theta + k_y < 0\}) \right] \leq \epsilon \tag{3.18}$$

with probability at least $1 - \delta$.

The statement made in the theorem above tells us that the worst case set of constraints (hence the sup) that are violated by sampling using distribution ψ happens rarely, ϵ, with probability at least $1 - \delta$. Note that the sample size needed m is dependent on the size of the vector θ, $|\theta| = n$.

The second source of error considered is how far is the approximation the RLP (3.15) is to the original ALP (3.11). A simplified version of the main theorem of De Farias and Van Roy [20] is given.

Theorem 3.2.5. *(Theorem 3.1 from De Farias and Van Roy [20])*

Let $\epsilon, \delta \in]0, 1[$. Let V be a Lyapunov function. Let $\kappa_{u^*,V,\mathcal{N}}$ be a constant which is a function of the optimal discounted state distribution u^*, The Lyapunov function V and a set $\mathcal{N}$. Let $\mathcal{X}$ be a set of m state-action pairs sampled independently according to the distribution $\psi_{\pi^*,V}(s, a)$ (the distribution is a function of the optimal policy π^* and V) where

$$m \geq \frac{16|\mathcal{A}|\kappa}{(1-\gamma)\epsilon} \left(n \ln \frac{48|\mathcal{A}|\kappa}{(1-\gamma)\epsilon} + \ln \frac{2}{\delta} \right)$$

Let $\tilde{\theta}$ be an optimal solution to the ALP that is in $\mathcal{N}$, and let $\hat{\theta}$ be an optimal solution of the corresponding RLP. If $\tilde{\theta} \in \mathcal{N}$ then, with probability at least $1 - \delta$, we have

$$||v^* - X\hat{\theta}|||_{1,\mu} \leq ||v^* - X\tilde{\theta}|||_{1,\mu} + \epsilon ||v^*||_{1,\mu}.$$

This rather convoluted theorem manages to bound the performance of the RLP in relation to the ALP. Combining this result with Theorem (3.2.3), we can bound the

performance of the RLP as,

$$||v^* - X\hat{\theta}||_{1,\mu} \leq \frac{2}{1-\gamma} \min_{\theta} \|v^* - X\theta\|_{\infty} + \epsilon||v^*||_{1,\mu}. \tag{3.19}$$

with probability at least $1-\delta$ for $m \geq \frac{16|\mathcal{A}|\kappa}{(1-\gamma)\epsilon}\left(n \ln \frac{48|\mathcal{A}|\kappa}{(1-\gamma)\epsilon} + \ln \frac{2}{\delta}\right)$.

Nonetheless, The ALP (3.11) and RLP (3.15) still present a few drawbacks. Despite the results of both theorem (3.2.5) and (3.2.3), we list a few of the drawbacks of the ALP and RLP which prevent them from truly being useful.

1. Both theorems involve the knowledge of the optimal discounted state distribution, u^* (3.10). This quantity is as difficult to find as v^* since it forms the basis of the dual LP-MDP (3.6). Computing these bounds are intractable in practice and therefore hinder their applicability.

2. Any practical implementation of the ALP or the RLP are still dependent on the size of the state space, $|\mathcal{S}|$. These approaches still require a starting state distribution, μ, and complete knowledge of the environment, such as the transition dynamics $P(s'|s,a)$, which are quantities of magnitude $\mathcal{O}(|\mathcal{S}|)$. Hence any practical implementation would require $\mathcal{O}(|\mathcal{S}|)$ storage.

3. The results from theorem (3.2.5) and (3.2.3) are relative bounds. Theorem (3.2.3) bounds the solution to the ALP relative to the optimal linear function approximation. Theorem (3.2.5) bounds the solution to the RLP solution relative to that of the ALP. These bounds may still be very loose and uninformative if the relative quantities they are compared to are very large.

4. This approach is only viable in the DP (planning) setting since knowledge of the environment is required. There's aren't any straightforward ways to extend this approach to the RL settings. Any attempt to do so, such as creating an approximate model of the environment, would introduce another source of possible error.

5. Extracting the global optimal policy from the ALP (or the RLP) requires $\mathcal{O}(|\mathcal{S}| \times |\mathcal{A}|)$ operations. Therefore, mitigating the impact of these suggested improvements.

These drawbacks provide a significant barrier for utilising the ALP in practice on large-scale MDPs and, as a result, the ALP hasn't garnered much attention when compared to other function approximation approaches such as policy gradient methods [64]. However, the theoretical guarantees given by this approach hint at the potential for LPs to solve some of the issues which plagues the theory of RL (Chapter 1). We now discuss two more extensions in this section which attempt to overcome some of these drawbacks in hopes of making this approach more practical.

3.2.2 Non-parametric Approximate Linear Program

Pazis and Parr [51] consider a non-parametric approach to the ALP that improves several aspects of De Farias and Van Roy [21]. The authors consider the value function LP (3.4) with the additional constraint of restricting the value function to a specific family of functions, $\mathcal{M}$.

$$\begin{aligned} \underset{\tilde{v}}{\text{minimize}} \quad & \sum_s \mu(s)\tilde{v}(s) \\ \text{subject to} \quad & \tilde{v}(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a)\tilde{v}(s') \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \\ & \tilde{v} \in \mathcal{M} \end{aligned} \tag{3.20}$$

Notice that when $\mathcal{M} = span(x)$ (where x is the feature vector) the optimization problem above (3.20) becomes equivalent the ALP (3.11) [21]. However, this formulation allows for arbitrary constraints on $\tilde{v}$ via $\mathcal{M}$ so long as they can be implemented through linear constraints (so that the resulting optimization problem remains a linear program). Consequently, the authors make the assumption that the optimal value function is Lipschitz continuous, with Lipschitz constant L, and impose this as a con-

straint to (3.20) via

$$|v^*(s) - v^*(s')| \leq L_{v^*} \cdot d(s, s'), \quad \forall s, s' \in \mathcal{S}$$

where $d(s, s') = ||k(s) - k(s')||$ with $k(s)$ some mapping from the state space to a normed vector space (i.e. some notion of distance between two states). If we denote by $\mathcal{M}_L$ to be the set of functions that are L-Lipschitz. The constraint $\tilde{v} \in \mathcal{M}_L$ can be enforced via linear constraints:

$$\tilde{v}(s) \geq \tilde{v}(s') - L_{\tilde{v}} \cdot d(s, s'), \quad \forall s, s' \in \mathcal{S} \tag{3.21}$$

If the Lipschitz assumption is true for the optimal value function (with Lipschitz constant say L_{v^*}) and we choose the correct Lipschitz constant, $L_{\tilde{v}} = L_{v^*}$, then including constraints (3.21) to (3.20) won't affect the solution, $\tilde{v} = v^*$. However, in practice we don't know L_{v^*} and the number of constraints are too large. The authors therefore limit themselves to a set $\tilde{S} \subseteq \mathcal{S}$ for which they enforce the Bellman recurrence constraints and the Lipschitz constraints. This leads to the following LP,

$$\begin{aligned} \underset{\tilde{v}}{\text{minimize}} \quad & \sum_{s \in \tilde{S}} \tilde{v}(s) \\ \text{subject to} \quad & \tilde{v}(s) \geq r(s, a) + \gamma \sum_{s' \in \tilde{S}} P(s'|s, a)\tilde{v}(s'), \quad \forall s \in \tilde{S},\ \forall a \in \mathcal{A} \\ & \tilde{v} \in \mathcal{M}_{L_{\tilde{v}}}(\tilde{S}), \end{aligned} \tag{3.22}$$

Where $\mathcal{M}_{L_{\tilde{v}}}(\tilde{S})$ denotes that the smoothness constraints are only enforced on the states in $\tilde{S}$. This LP is known as the non-parametric approximate linear program (NP-ALP). The optimization problem above has a number of variables and an objective function of size $|\tilde{S}| < |\mathcal{S}|$ and a set of constraints of size $|\tilde{S}| \times |\mathcal{A}|$.

This approach has the added benefit of being able to extrapolate to unseen states by leveraging the smoothness constraints (a notion of generalization). Let t be a unknown state when solving the NP-ALP (3.22), i.e. $t \notin \tilde{S}$,

we may estimate $\tilde{v}(t)$ by identifying a constrained state $s \in \tilde{S}$ that maximizes the associated Lipschitz constraint:

$$\tilde{v}(t) = \max_{s \in \tilde{S}} \{\tilde{v}(s) - L_{\tilde{v}} d(s,t)\}$$

Finding the state that maximizes this quantity would translate to the value that state t would have been assigned by the LP (3.22) if the smoothness constraint on t had been included (i.e. $\tilde{v}(s) - L_{\tilde{v}} \cdot d(s,t),\ \forall s \in \tilde{S}$).

The smoothness constraints can also aid in action selection. Since the LP is reduced to only a subset $\tilde{S}$, we might not have an action a for each state s such that the Bellman constraint associated with state-action pair (s,a) holds with strict equality (as we would expect when solving the full LP-MDP). States that have this equality property are called *basic* whereas those that don't are called *non-basic* by the authors. A basic state, s, has a corresponding action, a, such that $\tilde{v}(s) = \tilde{q}(s,a)$, which signifies a non-zero dual variables (see section 3.1). Action selection for a basic state, s, translates simply to choosing the action, a, corresponding to the non-zero dual variable for that state. However, for non-basic states and unseen states alike, we can deduce an action by finding the corresponding basic state, s, that maximizes the smoothness constraint. If a basic state s bounds the value of a non-basic state t by $\tilde{v}(t) \geq \tilde{v}(s) - L_{\tilde{v}} \cdot d(s,t)$, it also implicitly bounds $\tilde{q}(t,a)$. The predicted optimal action at t will therefore be the same as the one at the action at state s which maximizes the smoothness constraint (i.e. $\pi(a|s) = \pi(a|t) = 1$) since the bound from all other states are lower, implying a lower q values. This demonstrates a way to derive policies for all states and therefore can be used for large, even infinite state spaces. Additionally, action selection relies only on previously seen actions (in some training set for example). Therefore, this approach is independent of the size of the action space and can also be extended to large and infinite actions spaces.

The following theorem attempts to bound the error of the NP-ALP.

Theorem 3.2.6. *(Theorem 5.4. from Pazis and Parr [51])*

Define $\epsilon = \frac{2}{1-\gamma} \min_{v \in \mathcal{M}(L_{\tilde{v}})} \|v - v^*\|_\infty$, *let* $\tilde{v}$ *be the solution to the non-parametric ALP and let* μ *be a starting state distribution,* $\sum_{s \in \mathcal{S}} \mu(s) = 1$, *then,*

$$\|\tilde{v} - v^*\|_{1,\mu} \leq \epsilon + 2\frac{d_{\max} \cdot (L_{v^*} + L_{\tilde{v}})}{1-\gamma}$$

Where $d_{\max}$ is the maximum distance from a non-sampled state to the closest sampled state. Notice that the error term ϵ is very similar to the error bound of De Farias and Van Roy [21] (Theorem 3.2.3). When $\mathcal{M} = span(x)$ the bound is exactly the same. The additional term $d_{\max}(L_{v^*} + L_{\tilde{v}})$ bounds the violation of Bellman constraints missing in the sampled LP.

To summarize this approach, the NP-ALP [51] improves upon the shortcoming of the ALP [21] by extending it to possibly infinite state and action spaces and extending the hypothesis class of approximate solutions beyond linear functions to any Lipschitz function, through the added Lipschitz continuity constraints. The authors also provide intuitively interpretable bounds on performance which build upon the bounds of the ALP. However, a few difficulties remain:

1. The Lipschitz continuity assumption might be limiting. There are no clear way of knowing apriori if this assumption would hold for an arbitrary MDP and might require domain specific (expert) knowledge.

2. The performance of this approach relies heavily on the hyper-parameters introduced such as the distance metric $d(s, s')$ and the Lipschitz constant $L_{\tilde{v}}$. Moreover, the theorem provides bounds on the approximation to optimal value function v^* but it isn't immediately apparent whether this would translate to a near-optimal policy.

3. This approach is still incompatible with the RL setting since we require knowledge of environment, namely $P(s|s,a)$ and $r(s,a)$. Any model-based attempt to estimates these parameters could negatively impact the performance and guarantees of this approach.

The NP-ALP takes a step in the right direction and provides many advantages over the ALP, both in terms of practical implementation and theoretical guarantees using a more expressive class of function to approximate the value function. In the next chapter (4), we build upon some of the intuitive concepts exhibited with the NP-ALP, namely the notion of a distance metric and non-parametric representations, to introduce an LP approach using kernel-based methods which can be derived from Ormoneit and Sen [50].

4

Kernel-Based Linear Program

In this chapter, we demonstrate that the kernel-based approximate value iteration algorithm proposed by Ormoneit and Sen [50] can give rise to a kernel-based LP formulation. This approach is non-parametric, fully compatible within the RL setting and suitable to infinite state spaces.

The two main issues when trying to implement an LP in the RL setting are: removing the dependency on the transition dynamics and scaling to large state spaces. The two previously studied approaches (the ALP from section 3.2.1 and the NP-ALP from section 3.2.2) address the second issue without ever directly discussing how these can be extended to unknown environments. One could, however, build an approximate model of the environment using samples (a model-based approach) and compose the error bounds previously seen (Theorems 3.2.5 and 3.2.6) with bounds on model error such those of Ravindran and Barto [54] or Jiang et al. [30]. Instead, we opt to follow a different approach, that of Kernel-Based RL [50].

We first briefly summarize Kernel-Based RL [50] (section 4.1) and illustrate the favorable theoretical guarantees that accompany it. We then derive a kernel-based LP and experimentally validate this approach through an experiment on the Cart-Pole environment [15] (section 4.2). Finally, we concisely summarize and compare the different LP approaches discussed so far.

4.1 Kernel-Based Reinforcement Learning

This section briefly describes the work of Ormoneit and Sen [50]. The intuition behind their approach is to produce an approximation for the one-step Bellman operator $\mathcal{T}^\pi$ (discussed in chapter 2, section 2.1.1.2) using a kernel function. Instead of approximating the operator for a given policy π, the authors approximate the Bellman operator for each unique, fixed action $a \in \mathcal{A}$, i.e. they approximate $\mathcal{T}^a$ by $\hat{\mathcal{T}}^a$ such that $\hat{\mathcal{T}}^a \approx \mathcal{T}^a$. Notice that this operator is useful in that it can be used to describe the relation between the state-value function v and the action-value function q.

$$q_a^\pi = \mathcal{T}^a v^\pi \tag{4.1}$$

Where q_a^π is a vector of the action-values associated only to action a. This operator can also help express the Bellman optimality operator $\mathcal{T}^*$ (equation 2.14),

$$\mathcal{T}^* v = \max_a \mathcal{T}^a v = \max_a q_a \tag{4.2}$$

Let us denote a new operator T as the max operator. Then, we may rewrite the Bellman optimality equations (equations 2.12, 2.11) as,

$$q_a^* = \mathcal{T}^a T q_a^* \tag{4.3}$$

Using equation (4.3), and an approximate one-step Bellman operator $\hat{\mathcal{T}}^a$, we may derive approximate Bellman equations as $\hat{q}_a = \hat{\mathcal{T}}^a T \hat{q}_a$. This is the approach taken by Ormoneit and Sen [50] and utilised to estimate an approximately optimal policy.

To produce an approximate one-step Bellman operator, $\hat{\mathcal{T}}^a$, the authors of [50] motivate their approach by first considering an approximation based on partitioning the state space. Consider for example a finite number of partitions $B_1, \ldots, B_N$. If the rewards and transition probabilities are, in fact, constant along these partitions then, if sufficient historical data is available, we can approximate the transitions probabilities between partitions, $P(S_{t+1} \in B_j | S_t \in B_i, A_t = a)$, arbitrarily well. We may

then recover the value function for the MDP associated with these new transitions and solve for the optimal policy using the techniques discussed in section 2.1.
In principle, we may apply this approach to recover the value function for any arbitrary MDP by using finer and finer partitions which corresponds to constructing a sequence of piece-wise constant approximations of v. Taking this approach to the extreme, we would intuitively like to replace these hard partitions by *smoothing* methods which define "membership" through a some notion of distance, a *weighted kernel*.

Consider data collected into separate sets based on the actions. Let S^a be a historical dataset of times when action a was taken, i.e. $S^a = \{(x_s, y_s^a) | s = 1, \ldots, m\}$ with x_s the feature vectors for state s and y_s^a the feature vector associated to the next state transitioned to by taking action a in state s. The action-value function can now be approximated using kernel averaging. The authors define a kernel function $k_{S^a,b}$ and approximate the action value, q as:

$$q(x,a) = \mathbb{E}\left[r(x,a) + \gamma v(S_{t+1}) | S_t = x, A_t = a\right] \tag{4.4}$$

$$\approx \hat{\mathcal{T}}^a v(x) \tag{4.5}$$

$$= \sum_{(x_s, y_s^a) \in S^a} k_{S^a,b}(x_s, x)\left[r(x_s, a) + \gamma v(y_s^a)\right] \tag{4.6}$$

The kernel function $k_{S^a,b}$ is determined by a dataset, S^a, a hyper-parameter, b, and a univariate, non-negative "mother kernel" ϕ^+,

$$k_{S^a,b}(x_s, x) = \frac{\phi^+\left(\frac{||x_s - x||}{b}\right)}{\sum_{(x_u, y_u^a) \in S^a} \phi^+\left(\frac{||x_u - x||}{b}\right)} \tag{4.7}$$

In practice, the authors implement ϕ^+ as a Gaussian kernel where b becomes the standard deviation (referred to as the *bandwidth*). Intuitively, the approximation (4.6) uses a weighted average of previously seen data to estimate the expected return. The weighting is determined by how similar the data is to the state of interest x, using the kernel function. When using a Gaussian kernel, $\mathcal{N}(0,1)$, points closer to the mean, 0, will have higher weight. The point $(||x_s - x||/b)$ will be closer to zero if

$||x_s - x|| \to 0$, therefore having a higher influence on the approximation (4.6). The *bandwidth* parameter plays an important role in determining the threshold for inclusion/exclusion of points from the neighborhood.

Having defined a way to compute the action-value (equation 4.6), the authors then proceed to solve for the approximately optimal action-values by following the principal of *policy iteration* (section 2.1.1.3) and implementing an approximate value iteration algorithm (Algorithm 2), i.e. through successive applications of the approximate optimal Bellman operator:

$$\hat{q}_a \leftarrow \hat{\mathcal{T}}^a T \hat{q}_a \tag{4.8}$$

It's important to note that $\hat{\mathcal{T}}^a$ is only defined for states previously seen in the dataset S^a. Therefore, we only evaluate $\hat{q}_a \approx q_a^*$ for states, y_s^a, already seen in the dataset and **not** over all states. In doing so, we remove the dependency on the size of the state space from the solution method. Also, the constructed approximate one-step Bellman operator $\hat{\mathcal{T}}^a$ acts as an approximate model for the underlying MDP, therefore we no longer rely on knowledge of the dynamics of the MDP and instead follow a model-based approach. These two concepts directly address the issues previously discussed when implementing LPs in the RL setting and are essential observations for the LP we derive in the next section (section 4.2).

We now discuss the theoretical guarantees associated with this approach. The main results establish that the kernel-based approximate value iteration method (equation 4.8) converges to a *unique* fixed-point and, given appropriate conditions, that this method is *consistent*, meaning that additional training data always improves the quality of the estimate. This implies that the estimate will eventually lead to **globally optimal** performance. These results are in stark contrast to essencially locally optimality results when using parametric models [65] and seems

to overcome one of the main issues with RL, the instability and lack of theoretical guarantees on convergence when using function approximation and off-policy learning [13] [68] [5]. We explicitly state these theorems as follows:

Theorem 4.1.1. *(Theorem 1 from Ormoneit and Sen [50])*

The approximate value iteration method converges to a unique fixed point.

Theorem 4.1.2. *(Theorem 2 from Ormoneit and Sen [50])*

Let $b(m)$ be any admissible shrinkage rate used to evaluate the random operator $\hat{\mathcal{T}}^a$. Let $\hat{q}$ be an approximation for q^* using approximate value iteration. Then,

$$\|\hat{q} - q^*\|_\infty \xrightarrow{P} 0 \text{ as } m \to \infty.$$

Where an admissible shrinkage rate for the bandwidth parameter, $b(m)$, correspond to the condition that $b(m)^{d+1}\sqrt{m} \to \infty$ and $b(m) \to 0$ as $m \to \infty$ (with d the dimension of the state space). This condition (Lemma 2 from Ormoneit and Sen [50]) ensures that the bandwidth parameter decreases slowly enough so as to not incur a large variance in the estimate. Moreover, $\xrightarrow{P}$ signifies convergence in probability. This theorem assumes that the reward r and value functions q^* and v^* are smooth (i.e. Lipschitz continuous). What this theorem tells us is that, as the number of observed transitions $m \to \infty$, the solution to the equation $\hat{q}_a = \hat{\mathcal{T}}^a T \hat{q}_a$ should deviate arbitrarily little from q^*. Finally, the following theorem establishes the convergence of the policy derived from this method to the optimal policy.

Theorem 4.1.3. *(Theorem 4 from Ormoneit and Sen [50])*

By using the random operator $\hat{\mathcal{T}}^a T\hat{q}$ and an admissible shrinkage rate to approximate the true action-value function, $\mathcal{T}^a T q^*$, the probability of choosing a sub-optimal action converges to zero as the number of samples in each data set S^a goes to infinity.

4.2 Deriving a Kernel-Based LP

To obtain a LP from the Kernel-Based RL approach [50], we can compare the two. Consider the one-step Bellman constraints of the LP-MDP (3.4),

$$v \geq r^a + \gamma P^a v, \quad \forall a \in \mathcal{A} \tag{4.9}$$

The right-hand side of the above inequality is nothing more than the Bellman operator defined in the previous section (section 4.1), $\mathcal{T}^a$, i.e. $\mathcal{T}^a v = r^a + \gamma P^a v$. Therefore, using the same approximate one-step Bellman operator defined in section 4.1, $\hat{\mathcal{T}}^a$, we may approximate these constraints by:

$$\hat{v} \geq \hat{\mathcal{T}}^a \hat{v}, \quad \forall a \in \mathcal{A} \tag{4.10}$$

Which is only defined over the dataset collected S^a, much in the same way as (4.8), with $\hat{v}$ symbolizing the estimated value for v^*. This can written in component form (following equation 4.6) as :

$$\hat{v}(x) \geq \sum_{(x_s, y_s^a) \in S^a} k_{S^a,b}(x_s, x) \left[r(x_s, a) + \gamma \hat{v}(y_s^a) \right] \tag{4.11}$$

Having said this, we may directly substitute the above constraints into the LP-MDP, to obtain a new LP:

$$\begin{aligned} \underset{\hat{v} \in \mathbb{R}^{m \times M}}{\text{minimize}} \quad & \sum_{\forall a} \sum_{y_s^a \in S^a} \hat{\mu}(y_s^a) \hat{v}(y_s^a) \\ \text{subject to} \quad & \hat{v}(y_s^a) \geq \sum_{(x_s, y_s^{a'}) \in S^{a'}} k_{S^a,b}(x_s, y_s^a) \left[r(x_s, a') + \gamma \hat{v}(y_s^{a'}) \right], \ \forall a' \in \mathcal{A}, \ \forall s \in \{1, \ldots, m\} \\ & \hat{v}(y_s^a) \text{ unconstrained} \end{aligned} \tag{4.12}$$

Where the datasets for each action S^a are of similar size, $|S^a| = m$. This LP, which we call the KB-LP, has $m \times |\mathcal{A}|$ variables and $m \times |\mathcal{A}|$ constraints which refer to the number of data points collected. We also introduce a modified starting state distribution $\hat{\mu}$, defined over all collected data points.

Although the KB-LP may seem overly elaborate, it manages to tackle both previously stated issues when using LPs in the RL setting. Firstly, this approach removes the dependency on the environment dynamics, namely knowledge of the transition distribution and rewards. This is done through the implicit approximate model of the underlying MDP constructed using the approximate one-step Bellman operator $\hat{\mathcal{T}}^a$. Second, the size of the LP is no longer dependent on the size of the state space $|\mathcal{S}|$. This dependency is replaced by a dependency on the number of samples, m. Although the number of samples, m, may grow beyond the size of the state space, we argue that this substitution is beneficial since m is something we typically have control over. Therefore, this allows the user to have explicit control over the impact of the *curse of dimensionality* on their solution by varying m with respect to the resources available and the specifics of an MDP of interest.

This kernel-based approach enables policy evaluation for unseen states, x.

$$\begin{aligned}\hat{v}(x) &= \mathcal{T}\hat{\mathcal{T}}^a\hat{v} \\ &= \max_a \sum_{(x_s,y_s^a)\in S^a} k_{S^a,b}(x_s,x)\left[r(x_s,a)+\gamma\hat{v}(y_s^a)\right]\end{aligned} \tag{4.13}$$

Which allows this approach to be used in infinite state spaces. Given a new unseen state x, we may retrieve our estimated optimal policy $\hat{\pi}^*$ with,

$$\hat{\pi}(x) = \arg\max_a \sum_{(x_s,y_s^a)\in S^a} k_{S^a,b}(x_s,x)\left[r(x_s,a)+\gamma\hat{v}(y_s^a)\right] \tag{4.14}$$

Which is the greedy policy associated with the approximately optimal action-values $\hat{q}$ produced through solving the KB-LP (4.12).

Theoretically, this method has the same guarantees as kernel-based RL (section 4.1). The KB-LP (4.12) solves the same underlying MDP as the approximate value iteration algorithm proposed by Ormoneit and Sen [50] (equation 4.8). Therefore, the theory with respect to the resulting approximate solution $\hat{v}$ and $\hat{q}$ also hold for the solution to the KB-LP. Namely, the convergence of the KB-LP to a unique

fixed-point and the consistency of this solution: convergence to the optimal solution as the number of data points collected tends towards infinity (Theorems 4.1.1, 4.1.2 and 4.1.3) provided the bandwidth parameter decreases adequately (this assume Lipschitz continuous value functions and rewards) hold for the KB-LP.

Experimental validation. To further support the proposed KB-LP and provide a proof of concept for this approach, we empirically validate its performance on a classic benchmark task in RL, the Cart-Pole environment[1]. The Cart-Pole problem consists of balancing a pole upright on a cart. The cart may move laterally in either direction by applying a fixed force. The state of the environment is defined by 4 continuous variables (the features): the cart's lateral position x, the cart velocity $\dot{x}$, the pole angle θ and the pole's angular velocity $\dot{\theta}$. There are two discrete actions available to the agent: move right or move left. A reward of $+1$ is given for every time-step the pole remains upright with a pole angle larger than 12 degrees. The episode ends when the pole either falls or the cart exceeds the left or right side limits of the environment. The environment will also terminate after 200 time-steps have been reached. The maximum return attainable in this environment is therefore 200. For our experiments, we ran both the kernel-based approximate value iteration algorithm of Ormoneit and Sen [50] and the KB-LP using a normalized radial basis function as our kernel. To evaluate these methods, we average the performance obtained while running the greedy policy derived from their solutions, $\hat{v}_{\text{KB-LP}}$ and $\hat{v}_{\text{VI}}$ (equation 4.14) over 1000 episodes. Our experiments are run using 3 different setups:

- The first setup (Figure 4.1(a) and 4.1(b)) consists of learning using randomly collected data. The dataset S^a for each action a is created by generating random roll-outs (trajectories) of the Cart-Pole environment and storing the transitions and rewards obtained each time we take action a. We do this for 1000 observations of each action. Therefore, $|S^a| = 1000$ for each of the two actions. The

[1]Code: https://github.com/NadeemWard/kernel-based_RL

results in these figures are obtained using **the same** randomly generated dataset for both methods (KB-LP and the approximate value-iteration approach).

- In the second setup, we investigate how these methods scale as we increase the amount of data available. The setup is similar to the first except that we obtain **different** randomly generated datasets for each method and we increase the amount of data available for each action to 2000 samples, i.e. $|S^a| = 2000$ for each action. These experiments correspond to Figure 4.1(c) and 4.1(d).

- In our last setup (Figure 4.1(e) and 4.1(f)), we consider first deriving a policy by KB-LP and approximate value iteration using **different** random trajectories of size 1000, $|S^a| = 1000$, for each method. We then use this policy to generate 1000 new samples for each action. We call this data "guided". We then retrain our models on all 2000 samples consisting of both the random and guided data, $|S^a| = 2000$.

The results are visualized in Figure (4.1) for different values of the bandwidth hyper-parameter and $\gamma = 0.99$. The dark blue line is the average performance of the method over the 1000 episodes and the light blue region represents one standard deviation from the mean. We see that both methods have identical performance when trained on the same dataset of randomly generated trajectories and obtain the maximum reward possible in this environment, 200, for selected values of the bandwidth hyper-parameter (Figure 4.1(a) and (b)). This falls in line with what is expected since since both methods are essentially two different ways to solve the same underlying approximate MDP. When we increase the amount of data (Figure 4.1(c) and (d)), we see that performance improves over a larger set of the bandwidth values. Finally, when we augment the data using "guided" samples (Figure 4.1(e) and (f)) we see stable performance for bandwidth values which perform well (i.e. less variance for high performing bandwidth values). However, performance stagnates for poor performing bandwidth values. This can be explained by considering the fact that generating data using the

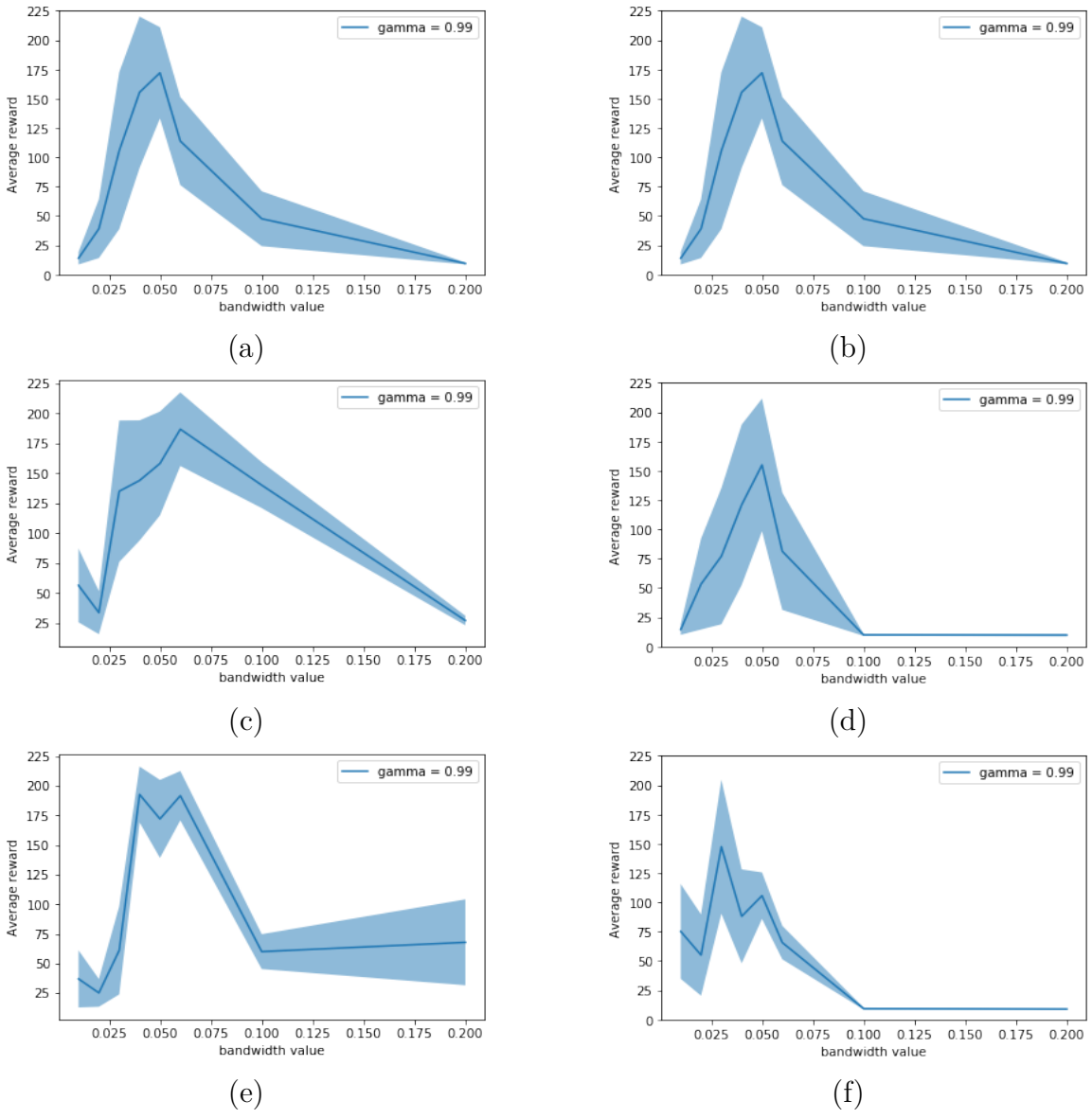

Figure 4.1: Average performance on Cart-Pole environment for different kernel bandwidth values. **Left**: KB-LP. **Right**: Kernel-Based Approximate Value Iteration [50]. **Top**: Data collected using 1000 random samples. **Middle**: Different data collected for each method using 2000 random samples. **Bottom**: Different data collected for each method using 1000 random samples and 1000 guided samples.

greedy policy will always yield similar trajectories. When performance is poor, these similar trajectories will provide low reward data and provide no information on how to improve in the environment. As a result, the agent will not know how to increase reward and continue to perform poorly. This can be seen as a case of premature *exploitation* in RL, where further *exploration* is necessary. Through these experiments, we provide empirical validation that the performance of the KB-LP is similar to the approximate value iteration algorithm proposed by Ormoneit and Sen [50].

Despite a few advantages, the KB-LP also presents some drawbacks. It is fully compatible with the RL setting, relying on sampled data and a kernel function rather than environment dynamics and the state space size. This approach is non-parametric and *always* converges to a unique solution given an increasing amount of data, a property that isn't guaranteed when using temporal difference learning with function approximation [13], [67]. However, we list a few significant drawbacks:

1. The results are heavily reliant on the kernel function used. The assumptions of smoothness and exploratory starts may also be limiting in some settings. Furthermore, we can't directly compare the theoretical results to those of the ALP [21] or the NP-ALP [51] to deduce the better approach.

2. Although the KB-LP removes the reliance of the LP-MDP on the size of the state space, $|\mathcal{S}|$. It introduces a reliance on the size of the data set, $m \times |\mathcal{A}|$, which can grow larger than $\mathcal{O}(|\mathcal{S}|)$. This necessitates further research to derive ways of reducing or aggregating the information.

To conclude this section, we provide Table (4.1) as a brief comparison of the different approaches discussed. Notice that since all of these methods are LPs, the run-time is polynomial in the number of variables and the number of constraints. Next, we move beyond directly trying to solve MDPs using an LP and elaborate on different ways that the LP formulation can be leveraged.

LP Approach	Hypothesis class	Number of variables	Number of constraints	Theoretical guarantees	Applicability to the RL setting
LP-MDP	any function in $\mathbb{R}^{\lvert\mathcal{S}\rvert}$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}\rvert \times \lvert\mathcal{A}\rvert$	exact solution	✗
ALP	linear functions in $\mathbb{R}^n$, $n << \lvert\mathcal{S}\rvert$	$\lvert\theta\rvert = n$	$\lvert\mathcal{S}\rvert \times \lvert\mathcal{A}\rvert$	within factor $\frac{2}{1-\gamma}$ of optimal linear approximation	✗
RLP	linear functions in $\mathbb{R}^n$, $n << \lvert\mathcal{S}\rvert$	$\lvert\theta\rvert = n$	$\lvert\mathcal{X}\rvert = m$, $m \geq \mathcal{O}\left(\frac{\lvert\mathcal{A}\rvert \ln \lvert\mathcal{A}\rvert}{(1-\gamma)\epsilon}\right)$	ϵ-close to ALP solution w.h.p.	✗
NP-ALP	non-parametric	$\lvert\tilde{\mathcal{S}}\rvert << \lvert\mathcal{S}\rvert$	$\lvert\tilde{\mathcal{S}}\rvert \times \lvert\mathcal{A}\rvert + \lvert\tilde{\mathcal{S}}\rvert$	$\frac{2}{1-\gamma}$ close to optimal lipschitz function + highest distance, $d_{\max}$	✗
KB-LP	non-parametric	$m \times \lvert\mathcal{A}\rvert$, with $m = \lvert S^a\rvert$	$m \times \lvert\mathcal{A}\rvert$, with $m = \lvert S^a\rvert$	converges to unique optimal solution as $m \to \infty$	✓

Table 4.1: Comparison table of different LP approach to RL.

5

Other Directions for LP in RL

This chapter discusses how the LP formulation for solving MDPs, the LP-MDP (equation 3.4), has been leveraged in other ways to create new algorithms, insights and theoretical guarantees. The recent approaches discussed here offer a glimpse at the future potential for using the LP formulation to further advance the field of RL.

Specifically, we first study the approach of Wang et al. [72] for deriving dual representation equivalent methods to those used in RL, described earlier in chapter 2). Second, we examine how the Lagrangian of the LP-MDP has been used to design provably efficient algorithms [16]. Finally, we explore how Nachum et al. [42] and Neu et al. [47] have manipulated duality and convex regularization to justify commonly used algorithms and create new methods for the off-policy and fixed dataset settings in RL.

5.1 Dual Representations for Dynamic Programming

In Wang et al. [72], the authors attempt to re-implement classical RL and Dynamic Programming (DP) methods from the dual perspective (equation 3.6). They do so by maintaining explicit representations of the discounted state distributions (equations 3.10 and 3.9) as opposed to value functions. Consequently, they derive new algorithm which alleviate some of the issues with value function based approaches.

Before explaining this approach, we first introduce some notation. Let $\mathbf{P}$ be the matrix of probability distributions for all state-action pairs, with dimension $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|$, where each row corresponds to the probability distribution $P(\cdot|s,a)^T$. Let the reward vector, r, and policy vector, π, be of size $|\mathcal{S}||\mathcal{A}| \times 1$. We denote by $r_{(sa)}$ the sa^{th} entry in the vector r, i.e. $r_{(s,a)} = r(s,a)$. Similarly, let $\mathbf{P}_{(sa,:)}$ denote the sa^{th} row in the matrix $\mathbf{P}$, i.e. $\mathbf{P}_{(sa,:)} = P(\cdot|s,a)$. Furthermore, let Ξ be a block diagonal matrix of size $|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|$ where each block is the vector $\mathbb{1}^T$ of size $|\mathcal{A}| \times 1$.

$$\Xi = \begin{pmatrix} 1\cdots1 & & & \\ & 1\cdots1 & & \\ & & \ddots & \\ & & & 1\cdots1 \end{pmatrix}$$

Similarly, define Π to be a block diagonal matrix representation of policy π such that each block is the conditional probability of taking an action in a given state, $\pi(\cdot|s)^T$.

$$\Pi = \begin{pmatrix} \pi(\cdot|s_0)^T & & & \\ & \pi(\cdot|s_1)^T & & \\ & & \ddots & \\ & & & \pi(\cdot|s_{|S|}^T) \end{pmatrix}$$

Π is of size $|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|$. Using this notation, we may describe the $|S| \times |S|$ state transition matrix P^π as ΠP. Likewise, we may define the state-action to state-action transition distribution $P^\pi(s,a|s'a')$ as $P\Pi$. The dual LP-MDP (3.6) may be written as:

$$\begin{aligned} \underset{d}{\text{maximize}} \quad & d^T r \\ \text{subject to} \quad & \Xi d = (1-\gamma)\mu + \gamma \mathbf{P}^T d, \\ & d \succeq 0 \end{aligned} \tag{5.1}$$

The primal and dual variables can be written compactly using this notation. For example, Πr defines the expected reward over states, $r(s)$. Therefore, we

may write the value function as:

$$v = \sum_{i=0}^{\infty} \gamma^i (\Pi \mathbf{P})^i \cdot \Pi r \tag{5.2}$$

$$= \Pi r + \gamma \Pi \mathbf{P} v \tag{5.3}$$

Similarly, the action value function q may be written as:

$$q = \sum_{i=0}^{\infty} \gamma^i (\mathbf{P}\Pi)^i r \tag{5.4}$$

$$= r + \gamma \mathbf{P}\Pi q \tag{5.5}$$

Where the last lines in the above quantities (5.3), (5.5) are the Bellman recurrence equations (2.4), (2.6) in matrix form. The dual variables may also be written in this way,

$$u^T = (1-\gamma)\mu^T \sum_{i=0}^{\infty} \gamma^i (\Pi \mathbf{P})^i \tag{5.6}$$

$$= (1-\gamma)\mu^T + \gamma u^T \Pi \mathbf{P} \tag{5.7}$$

and

$$d^T = (1-\gamma)\nu^T \sum_{i=0}^{\infty} \gamma^i (\mathbf{P}\Pi)^i \tag{5.8}$$

$$= (1-\gamma)\nu^T + \gamma d^T \mathbf{P}\Pi \tag{5.9}$$

Where ν is an initial state-action distribution of size $|\mathcal{S}||\mathcal{A}| \times 1$ defined as $\nu = \Pi^T \mu$ (the policy π times the starting state distribution). The relationship between u and v can be written as $(1-\gamma)\mu^T v = u^T \Pi r$, and the relationship between d and q as $(1-\gamma)\nu^T q = d^T r$. All of these quantities can be solved for a given policy Π either by iterative policy evaluation (Algorithm 1) or by solving the matrix inverse (equation 2.8) as seen in chapter 2.

Modified dual variables are used to facilitate policy improvement in the dual domain. The authors of [72] use slightly different dual representations that

decouple them from their dependence on the initial state distribution. Instead of u, consider

$$M = (1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\Pi\mathbf{P})^i \tag{5.10}$$

$$= (1-\gamma)I + \gamma\Pi\mathbf{P}M \tag{5.11}$$

Where M is a matrix of size $|\mathcal{S}| \times |\mathcal{S}|$ which can also be expressed recursively (5.11). Similarly, consider H instead of d.

$$H = (1-\gamma)\sum_{i=0}^{\infty}\gamma^i(\mathbf{P}\Pi)^i \tag{5.12}$$

$$= (1-\gamma)I + \gamma\mathbf{P}\Pi H \tag{5.13}$$

With H a matrix of size $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|$. These matrices M and H represent the conditional discounted state distributions conditioned on either states or state-action pairs, respectively. The relationship between u and M can be expressed as $u^T = \mu^T M$ (Lemma 5 from [72]). This implies that the relationship between M and primal variables v can be expressed as $(1-\gamma)v = M\Pi r$. Similarly, $d^T = \nu^T H$ which implies that the relationship to q can be written as $(1-\gamma)q = Hr$. Also, since $v = \Pi q$ we may say that $M\Pi = \Pi H$.

Dual policy improvement can be carried out similar to policy improvement methods described in chapter 2. The dual form of the greedy policy update can be expressed in terms of the state-action transition matrix H for π.

$$a^*(s) = \arg\max_a \; q^*_{(sa)} = \arg\max_a \; H_{(sa,:)}r \tag{5.14}$$

Hence the new greedy policy can be expressed as $\pi'(a|s) = \mathbb{1}_{\{a=a*(s)\}}$. Implementing a dual version of policy iteration can therefore proceed in a similar way to the original policy iteration algorithm (Algorithm 2) by alternating between policy evaluation of M (solving the equation $M = (1-\gamma)I + \gamma\Pi\mathbf{P}M$) and policy improvement, using $a^*(s) = \arg\max_a H_{(sa,:)}r = \arg\max_a((1-\gamma)I_{(sa)} + \gamma\mathbf{P}_{(sa,:)}\Pi H)r =$

$\arg\max_a \mathbf{P}_{(sa,:)} M\Pi r$, until convergence.

Temporal difference learning methods can also be derived using dual variables M and H. Having established the relationship between the primal state value function v and the dual discounted state visitation distribution M, dual temporal difference learning methods can be derived analogously. For policy evaluation, recall the one-step TD update (2.22),

$$v(s) \leftarrow v(s) + \alpha\Big[r + \gamma v(s') - v(s)\Big] \tag{5.15}$$

which is mimicked for the dual matrix M as follows:

$$M_{(s,:)} \leftarrow M_{(s,:)} + \alpha\Big[(1-\gamma)\mathbf{1}_s^T + \gamma M_{(s',:)} - M_{(s,:)}\Big] \tag{5.16}$$

As for dual policy improvement and control using temporal difference learning, an analogous version of the *Sarsa* algorithm [58] may be derived using ϵ-greedy policy improvement (equation 5.14) and policy evaluation of the form:

$$H_{(sa,:)} \leftarrow H_{(sa,:)} + \alpha\Big[(1-\gamma)\mathbf{1}_{sa}^T + \gamma H_{(s'a',:)} - H_{(sa,:)}\Big] \tag{5.17}$$

Similarly, the Q-learning algorithm (Algorithm 3) update:

$$q(s,a) \leftarrow q(s,a) + \alpha\Big[r + \gamma \max_{a'} q(s',a') - q(s,a)\Big] \tag{5.18}$$

is replaced by a dual version,

$$H_{(sa,:)} \leftarrow H_{(sa,:)} + \alpha\Big[(1-\gamma)\mathbf{1}_{sa}^T + \gamma H_{(s'a'^*,:)} - H_{(sa,:)}\Big] \tag{5.19}$$

where a'^* is the greedy policy $a'^* = \arg\max_{a'} \; H_{(s'a',:)} r$.

Linear function approximation may also be implemented in the dual domain. Wang et al. [72] suggest a linear function approximation variant to the dual LP-MDP (3.6) following the example of De Farias and Van Roy [21] in the primal

case. Since the dual variable, d, is a probability distribution, the authors consider an approximation using a linear combination of predefined ("basis") distributions, $\Psi w \approx d$, and insure the resulting approximation remains a valid distribution. Where Ψ is a matrix of size $|S||A| \times k$ with each column forming a valid probability (basis) distribution over state-action pairs. To insure that the resulting solution remains a valid distribution, additional convex combination constraints are added to the parameter values w ($w \succeq 0$ and $w^T\mathbf{1} = 1$). The resulting linear approximate dual version of the ALP [21] (equation 3.11) is:

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & (\Psi w)^T r \\ \text{subject to} \quad & \Xi\Psi w \leq (1-\gamma)\mu + \gamma \mathbf{P}^T \Psi w, \\ & w \succeq 0,\ w^T\mathbf{1} = 1 \end{aligned} \tag{5.20}$$

Gradient based methods are yet another approach which can be presented in the dual domain. Generally, gradient based methods in the primal domain propose a parametric model $\hat{v}(s,w) \approx v^\pi(s)$ and minimize the mean square value error, $\overline{\text{VE}}$:

$$\overline{\text{VE}}(w) = \sum_{\forall s} \mu(s)\Big[v_\pi(s) - \hat{v}(s,w)\Big]^2 \tag{5.21}$$

This can be done using stochastic gradient updates. When using a linear function approximation, $v^\pi \approx x^T w$, the update becomes:

$$\begin{aligned} w &\leftarrow w - \frac{1}{2}\alpha\nabla\Big[v_\pi(s) - \hat{v}(s,w)\Big]^2 \\ &\leftarrow w + \alpha\Big(r(s) + \gamma\hat{v}(s',w) - \hat{v}(s,w)\Big)x^T_{(s,:)} \end{aligned}$$

Implementing this approach in the dual domain can follow a similar procedure. Consider an objective with dual representations of the form:

$$J = \frac{1}{2}||M - \hat{M}||^2_\mu \tag{5.22}$$

With $\hat{M}$ a convex combination of basis distributions of size $|S| \times |S|$, $\hat{M} = \Upsilon W \Gamma$ with the parameters now a matrix W of size $k \times k$ and Υ, Γ the predefined basis

distributions. The update equations for the weights in W can be written as:

$$W \leftarrow W + \alpha\delta\Delta \tag{5.23}$$

With δ the one step bootstrap estimate $\delta = 1 - \gamma + \gamma\hat{M}_{(s',s'')} - \hat{M}_{(s,s')}$ and Δ the gradient of the objective function J projected onto the space of feasible weights (to ensure a valid distribution). Δ can be obtained by solving the following quadratic program (QP):

$$\begin{aligned} \underset{\Delta}{\text{minimize}} \quad & \sum_{i,j} \Big(\Delta_{(i,j)} - D_{(i,j)}\Big)^2 \\ \text{subject to} \quad & \Delta\mathbb{1} = 0 \end{aligned} \tag{5.24}$$

Where D is the unconstrained gradient and Δ is the projection onto the constraint. There exists a closed form solution for this QP of the form,

$$\Delta = D - \frac{1}{k}D(\mathbf{1}\mathbf{1}^T) \tag{5.25}$$

Which therefore allows us to write the weight updates to W as:

$$W \leftarrow W + \alpha\delta D(I - \frac{1}{k}\mathbf{1}\mathbf{1}^T) \tag{5.26}$$

The different methods presented earlier demonstrate the validity of the dual representation as an alternative to the commonly used value function in RL. By replacing the value function with explicit representation of the discounted state distribution in commonly used RL methods, the authors demonstrate a viable alternative approach. Additionally, these approaches alleviates the worst-case unbounded divergence behavior present in value function based methods [5] since dual representations must remain valid distributions. This is enforced through constraints in the dual linear approximate LP-MDP (equation 5.20) and through projection in the gradient based method (equation 5.23). However, few challenges remain:

1. The computation complexity associated with ensuring the dual variables remain valid distribution may render the proposed methods intractable. It will lead to

additional computation cost which may be significant therefore hindering the applicability of this approach.

2. The dual variables present a new set of issues due to the size and sparsity of their representations. This may lead to slow convergence and numerically instability when learning.

5.2 Leveraging the Lagrangian

The LP formulation can be used as a starting point from which to derive new approaches based on convex optimization. Here, we discuss the approach of Chen and Wang [16] who use the Lagrangian of the LP-MDP (3.4) to derive a new algorithm with strong theoretical guarantees.

We can modify the LP-MDP to create a saddle-point problem. To do so, we first derive the Lagrangian of our problem of interest. Recall the primal LP-MDP (3.5),

$$\begin{aligned} \underset{v\in\mathbb{R}^{|\mathcal{S}|}}{\text{minimize}} \quad & \mu^T v \\ \text{subject to} \quad & (\gamma P^a - I)v + r^a \geq 0, \quad \forall a \in \mathcal{A} \\ & v \text{ unconstrained} \end{aligned}$$

The Lagrangian for this LP is of the form,

$$L(v, d) = \mu^T v + \sum_{a\in\mathcal{A}} d_a^T \left((\gamma P^a - I)v + r^a\right) \tag{5.27}$$

Where d_a is a vector of size $|\mathcal{S}|$ corresponding to the dual variables associated with action a, i.e. $d_a = [d(s_0, a), \ldots, d(s_{|\mathcal{S}|}, a)]^T$. We can write the primal LP as a saddle-point problem using the Lagrangian:

$$\min_{v\in\mathbb{R}^{|\mathcal{S}|}} \max_{d\succeq 0} L(v, d) = \min_{v\in\mathbb{R}^{|\mathcal{S}|}} \max_{d\succeq 0} \left\{\mu^T v + \sum_{a\in\mathcal{A}} d_a^T((\gamma P^a - I)v + r^a)\right\} \tag{5.28}$$

Chen and Wang [16] study a modification of the above saddle-point formulation. This modification augments the previous saddle-point problem (5.28) by restricting the primal and dual variables v and d to constrained sets.

$$\min_{v\in\mathbb{R}^{|\mathcal{S}|}} \max_{d\in\mathbb{R}^{|\mathcal{S}||\mathcal{A}|}} \left\{ \mu^T v + \sum_{a\in\mathcal{A}} d_a^T((\gamma P^a - I)v + r^a) \right\} \tag{5.29}$$

$$\text{subject to} \quad v \in \mathcal{V} \quad d \in X \cap \Delta \tag{5.30}$$

With $\mathcal{V} = \left\{v | v \geq 0, ||v||_\infty \leq \frac{R_{\max}}{1-\gamma}\right\}$, $X = \{d | \sum_{a\in A} d(s,a) \geq \mu(s),\ \forall s \in \mathcal{S}\}$ and $\Delta = \left\{d | d \geq 0, ||d||_1 = \frac{||\mu||_1}{1-\gamma}\right\}$. These sets describe easily derivable information about the solution. Notice for example, that the maximum reward an agent can receive in a discounted finite reward MDP is $\sum_{t=0}^{\infty} \gamma^t R_{\max} = \frac{R_{\max}}{1-\gamma}$. Therefore, it must be that $||v||_\infty \leq \frac{R_{\max}}{1-\gamma}$, the constraint described in set $\mathcal{V}$. Furthermore, the sets X and Δ can be derived from the dual LP constraints (3.6). Recall that the constraints are of the from $\sum_a d_a(I - \gamma P^{aT}) = \mu$. Since P^a is a probability matrix with values ≤ 1, we can directly infer $\sum_a d(s,a) \geq \mu(s)$, which corresponds to set X. Similarly, if we consider d^* to be the vector of non-zero valued optimal dual variables (of size $|\mathcal{S}|$ since there is only 1 non-zero dual variable per state), we may write $d^* = \mu + \gamma(P^{\pi^*})^T d^*$, with $d^* \succeq 0$ and $\mu \succeq 0$. From this, we may derive the following:

$$\|d^*\|_1 = e^T d^* = e^T \mu + \gamma e^T (P^{\pi^*})^T \mu^* = e^T \mu + \gamma e^T d^* = \|\mu\|_1 + \gamma \|d^*\|_1$$

Which implies that $\|d^*\|_1 = \frac{\|\mu\|_1}{1-\gamma}$ corresponding to constraint set Δ. These observation about the sets $\mathcal{V}$, X and Δ imply that the solution to the modified saddle point problem (5.29) is equivalent to that of the original problem (5.28).

The algorithm proposed by Chen and Wang [16] is concerned with a setting where the state space $\mathcal{S}$, action space $\mathcal{A}$, reward upper bound $R_{\max}$ and discount factor γ are known. The transition probabilities P and rewards function r are unknown and we assume access to a generative model of the dynamics of the environment, called a sample oracle, denoted $\mathcal{SO}$, which can be queried to get samples (i.e. given a state-

action pair (s, a), $\mathcal{SO}$ generates s' w.p. $P(s'|s, a)$ and reward r from $r(s', a, s)$).

To implement the proposed algorithm, we update running estimates of both the primal variables v and the dual variables d using stochastic gradients with projection, hence the name *Stochastic Primal-Dual* given to the method [16]. Several authors assimilate this approach to *Mirror-Descent* [9], which also takes gradient steps in the dual space [70] [8]. Per iteration, the algorithm samples a state-action (s, a) pair uniformly and receive a next state s' and reward r from $\mathcal{SO}$. The primal and dual variables are updates as,

$$v^{(k)}(s) \leftarrow \max\left\{\min\left\{v^{k-1}(s) - \beta\left(\frac{1}{|\mathcal{A}|} - d_a^{(k-1)}(s)\right), \frac{R_{\max}}{1-\gamma}\right\}, 0\right\} \tag{5.31}$$

$$v^{(k)}(s') \leftarrow \max\left\{\min\left\{v^{k-1}(s') - \gamma\beta d_a^{(k-1)}(s), \frac{R_{\max}}{1-\gamma}\right\}, 0\right\} \tag{5.32}$$

$$d_a^{k-\frac{1}{2}}(s) \leftarrow d^{k-1}(s, a) + \beta\left(\gamma v^{k-1}(s') - v^{k-1}(s) + r\right) \tag{5.33}$$

$$d^k \leftarrow \Pi_{X\cap\Delta} d^{k-\frac{1}{2}} \tag{5.34}$$

Where Π is the projection operator of the dual variables onto their constraint sets. These updates are a modification of the gradient for the modified saddle-point problem (5.29) with respect to the primal and dual variables. After termination, the dual variables are then averaged over time to provide an estimate to the optimal dual variables, $\hat{d}$, which can be used to produce a policy $\hat{\pi}(a|s) = \hat{d}_a(s)/\sum_a \hat{d}_a(s)$. Per iteration, the individual updates take constant time $O(1)$. The dual variables are then projected back into the space $X \cap \Delta$ at an additional cost of $O(|S| \times |A|)$ per iteration. Only the primal variables associated to the sampled s and s' ($v(s)$ and $v(s')$) and the associated dual variable of sampled action a, $d_a(s)$, are updated.

The authors prove PAC style [69] theoretical guarantees on the performance of their algorithm.

Theorem 5.2.1. *(Theorem 3 from Chen and Wang [16])*

For any $\epsilon > 0$, $\delta \in]0, 1[$, *the Stochastic Primal-Dual algorithm proposed with a number of iterations*

$$\Omega\left(\frac{|\mathcal{S}|^4|\mathcal{A}|^2 R_{\max}^2}{(1-\gamma)^6\epsilon^2}\ln\left(\frac{1}{\delta}\right)\right)$$

will produce an ϵ*-optimal policy* $\hat{\pi}$ *with probability at least* $1-\gamma$.

Where an ϵ-optimal policy π means $\max_{s\in\mathcal{S}}|v^{\pi}(s) - v^{*}(s)| \leq \epsilon$. This theorem was obtained by leveraging results in the optimization literature to analysing the duality gap between estimates v^k and d^k as the number of iterations grew. Bas-Serrano and Neu [8] study an extension to this approach which considers a linear relaxation to the primal and dual constraints of the saddle-point problem (5.28) in the average reward undiscounted setting. There, the authors provide guarantees on the performance of a policy derived from this linear relaxation using a Mirror-Descent type algorithm when access to the environment dynamics are known.

5.3 Duality and Convex Regularization in RL

The theory of convex optimization provides strong foundations for understand and justifying convex regularization [14]. As demonstrated by Neu et al. [47] and Nachum et al. [42], these insights can be leveraged for RL.

Recently, the works of Nachum et al. [44], [43], [42] explore the use of duality to derive new insights and algorithms for off-policy learning with a fixed dataset, $\mathcal{D}$. The first objective of this line of work is to reformulate the maximum-return problem in RL,

$$\max_{\pi} J(\pi) = (1-\gamma)\mathbb{E}_{s\sim\mu, a\sim\pi(.|s)}[q(s,a)] = \mathbb{E}_{(s,a)\sim d^{\pi}}\left[r(s,a)\right] \tag{5.35}$$

And make it amenable to fixed-dataset scenarios without access to the policy used to generate the data. In doing so, the reformulation would be *behavior-agnostic*. The

second objective is to derive feasible algorithms from this formulation.

The authors introduce an action value equivalent to the LP-MDP (3.4). This LP, called the Q-LP, is used for evaluating a fixed policy π.

$$\begin{aligned}\rho(\pi) = \underset{q\in\mathbb{R}^{|\mathcal{S}||\mathcal{A}|}}{\text{minimize}} \quad & \mathbb{E}_{\substack{s\sim\mu \\ a\sim\pi(a|s)}}[q(s,a)] \\ \text{subject to} \quad & q(s,a) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') q(s',a'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S} \\ & q(s,a) \text{ unconstrained},\end{aligned} \tag{5.36}$$

It's subsequent dual can be written as,

$$\begin{aligned}\underset{d}{\text{maximize}} \quad & \sum_{s,a} d(s,a) r(s,a) \\ \text{subject to} \quad & d(s,a) = (1-\gamma)\mu(s)\pi(a|s) + \gamma \sum_{s'}\sum_{a'} \pi(a|s) P(s|s'a') d(s',a'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S} \\ & d \succeq 0, \quad \forall a \in \mathcal{A}, s \in \mathcal{S}\end{aligned} \tag{5.37}$$

The dual problem (5.37) can be manipulated to yield a new objective function independent of the data generating policy. Notice that the dual constraint for the problem (5.37) are sufficient to completely define d. The objective function plays no part in determining the values of d. Therefore, the authors propose to replace it with a convex regularizer. In doing so, the authors introduce data dependent variables of interest, $d^{\mathcal{D}}$, which can later be manipulated to reformulate the objective function (5.35). The described approach can be written as,

$$\begin{aligned}\underset{d}{\text{maximize}} \quad & -D_f(d||d^{\mathcal{D}}) \\ \text{subject to} \quad & d(s,a) = (1-\gamma)\mu(s)\pi(a|s) + \gamma \sum_{s'}\sum_{a'} \pi(a|s) P(s|s'a') d(s',a'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S} \\ & d \succeq 0, \quad \forall a \in \mathcal{A}, s \in \mathcal{S}\end{aligned} \tag{5.38}$$

Where $D_f(d||d^{\mathcal{D}})$ is the f-divergence from the offline discounted state-action distribution, d, which is convex. The following Fenchel-Rockafellar dual [57] to this convex optimization problem (5.38) is unconstrained thereby simplifying it's optimization [44],

$$\min_q (1-\gamma)\mathbb{E}_{s\sim\mu,a\sim\pi(.|s)}[q(s,a)] + \mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[f_*(\gamma \sum_{s',a'} P(s',a'|s,a)q(s',a') - q(s,a))\right] \tag{5.39}$$

Where f_* is the Fenchel-Rockafellar dual. This problem is unconstrained, convex and no longer dependent on the knowledge of the data generation process d. We can compute this quantity for a given policy π thereby enabling policy evaluation in the fixed data setting. This line of reasoning is the main idea behind *DualDice* [44].

***AlgeaDice* [43] takes this one step further by tackling the problem of policy improvement**. Following similar steps to the above, the authors consider a different regularized dual problem (5.37),

$$\begin{aligned}
\underset{d}{\text{maximize}} \quad & \sum_{s,a} d(s,a)r(s,a) - D(d||d^{\mathcal{D}}) \\
\text{subject to} \quad & d(s,a) = (1-\gamma)\mu(s)\pi(a|s) + \gamma \sum_{s'}\sum_{a'} \pi(a|s)P(s|s'a')d(s',a'), \quad \forall a \in \mathcal{A}, s \in \mathcal{S} \\
& d \succeq 0, \quad \forall a \in \mathcal{A}, s \in \mathcal{S}
\end{aligned} \tag{5.40}$$

From which they derive a similar unconstrained objective which involves only the fixed data generating distribution $d^{\mathcal{D}}$ using Fenchel-Rockafellar duality. However, the resulting dual problem still only solves the policy evaluation problem. For policy improvement, we must maximize over all policy which can be evaluated using this newly formed Fenchel-Rockafellar dual problem thereby creating a bi-level optimization problem which has no guarantee of convexity.

Nachum et al. [42] also explore applying Fenchel-Rockafellar duality to a regularized version of the original dual LP-MDP (3.6). However, the resulting optimization prob-

lem, although convex, can't be easily manipulated to extract the optimal policy. We must solve a separate optimization problem to approximate π^* (bi-level optimization). Furthermore, it's isn't apparent whether the approximate v^* will yield near-optimal policy π^*.

Duality and convex regularization can be used to derive Entropy Regularized MDPs [47]. By modifying the dual LP-MDP (3.6) to incorporate a convex regularizer, Neu et al. [47] create a regularized MDP framework. The modified dual LP-MDP is of the form:

$$\max_{d\in\Delta} \tilde{J}_\eta(d) = \max_{d\in\Delta} \left\{ \sum_s \sum_a d(s,a) r(s,a) - \frac{1}{\eta} R(d) \right\} \tag{5.41}$$

Where Δ is the convex set of valid probability distribution in $\mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$. The term $R(d)$ is a convex regularizer (the authors study negative Shannon entropy and the negative conditional entropy) and η a learning rate. Using this convex optimization problem (5.41), the authors derive a framework for regularized MDPs which provides theoretical justification for regularization approaches in RL by linking them to concepts in convex optimization. More specifically, they demonstrate that TRPO [61] and the regularized policy gradient method of Mnih et al. [40] are approximate versions of Mirror Descent [9] and Dual Averaging [74] respectively.

Vieillard et al. [70] create similar links between RL and optimization. There, the authors demonstrate that Conservative Policy Iteration [31] is equivalent to the Frank-Wolfe algorithm [24], Mirror-Descent [9] is equivalent to Modified Policy Iteration [25] and Dual Averaging [45] is equivalent to Politex [1].

6

Conclusion and Future Work

This thesis attempts to study the usefulness of the LP approach to solving MDPs. By analyzing its progression over time, this work highlights why it has failed to garner traction in the past and why it still holds potential for future research in RL.

LPs are a viable way of solving exact MDPs with good theoretical guarantees. Using the LP formulation for solving MDPs (3.4) described by Puterman [53] is guaranteed to produce the optimal policy in a number of arithmetic operations polynomial in $|\mathcal{S}|$, $|\mathcal{A}|$ and B, the maximum number of bits needed to represent the model dynamics. Moreover, the dual LP-MDP (3.6) provides a unique perspective for solving MDPs from which new algorithms can be derived (Wang et al. [72]). Also, the LP formulation gives an insightful link to the notion of Successor Representations [18] and facilitates adding constraints when studying constrained MDPs [3].

However, scaling the LP approach is difficult to implement in practice and is unsuitable to the RL setting. Despite the works of De Farias and Van Roy [21] [20] which manage to extend the LP approach to the function approximation setting and provide theoretical guarantees on performance. Issues with its implementation in large state spaces along with theoretical guarantees dependent on quantities intractable to compute for specific problem instances make this approach unusable in most cases. In

addition, incorporating this approach into the RL setting is not straightforward and would necessitate constructing an approximate model of the environment, thereby introducing another source of approximation error. Further improvement such as the NP-ALP [51] (section 3.2.2) and the KB-LP (chapter 4) provide hope for more practical implementations but aren't quite there yet. Whether that be because they aren't fully adapt to the RL setting or because they don't process data efficiently.

Nonetheless, several recent innovative works have explored new directions which use the LP approach to pivot and create new algorithms and establish new connections between RL and optimization. Chen and Wang [16] exploit the Lagrangian of the LP formulation to derive a new algorithm based on convex-optimization which provides PAC-style guarantees on performance in a slightly modified RL setting. Neu et al. [47] use the dual LP (3.6) to derive a theory of regularized MDPs and provide theoretical justification to regularization methods used in RL. Nachum et al. [44], [43], [42] successively demonstrate how to manipulate LP duality to derive new formulas for the difficult off-policy and fixed dataset settings. These connections may be leveraged to extract new insights and guarantees.

The link between RL and convex optimization may continue to be developed to provide new insights, algorithms and theoretical guarantees which would be mutually beneficial to both areas of research. Some suggested promising directions for future research include:

- Algorithms such as those proposed by Chen and Wang [16] provide novel approaches in RL which maintain and update both value function and discounted state distribution estimates. Each can provide distinct information about the environment which may be aggregated to better balance the problem of exploration versus exploitation.

- Convex optimization algorithms are known to have problem dependent performance. It is important to study the circumstances that lead to the poor performance for a specific algorithm on a class of MDPs. This may lead to better understanding of the underlying structure and characteristics of an MDP which can be exploited [38].

- Finally, further exploration of LP duality in RL [42] may lead to effective ways of tackling issues in off-policy learning and fixed data and provide guarantees on performance.

Bibliography

[1] Yasin Abbasi-Yadkori et al. "POLITEX: Regret Bounds for Policy Iteration using Expert Prediction". In: ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 3692–3702. URL: http://proceedings.mlr.press/v97/lazic19a.html.

[2] Pieter Abbeel, Adam Coates, and Andrew Ng. "Autonomous Helicopter Aerobatics through Apprenticeship Learning". In: *I. J. Robotic Res.* 29 (Nov. 2010), pp. 1608–1639. DOI: 10.1177/0278364910371999.

[3] Eitan. Altman. *Constrained Markov decision processes*. English. Boca Raton; London: Chapman & Hall/CRC, 1999.

[4] Pierre-Luc Bacon. "Temporal Representation Learning". PhD thesis. McGill University, June 2018.

[5] Leemon Baird. "Residual Algorithms: Reinforcement Learning with Function Approximation". In: *In Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 30–37.

[6] Stefan Banach. "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales". fre. In: *Fundamenta Mathematicae* 3.1 (1922), pp. 133–181. URL: http://eudml.org/doc/213289.

[7] Andre Barreto et al. "Successor Features for Transfer in Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 4055–4065. URL: https://proceedings.neurips.cc/paper/2017/file/350db081a661525235354dd3e19b8c05-Paper.pdf.

[8] Joan Bas-Serrano and Gergely Neu. "Faster saddle-point optimization for solving large-scale Markov decision processes". In: ed. by Alexandre M. Bayen et al. Vol. 120. Proceedings of Machine Learning Research. The Cloud: PMLR, Oct. 2020, pp. 413–423. URL: http://proceedings.mlr.press/v120/serrano20a.html.

[9] Amir Beck and Marc Teboulle. "Mirror descent and nonlinear projected subgradient methods for convex optimization". In: *Operations Research Letters* 31.3 (2003), pp. 167–175. ISSN: 0167-6377. DOI: https://doi.org/10.1016/S0167-6377(02)00231-6. URL: http://www.sciencedirect.com/science/article/pii/S0167637702002316.

[10] R. Bellman, R.E. Bellman, and Rand Corporation. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. URL: https://books.google.com/books?id=rZW4ugAACAAJ.

[11] Richard Bellman. "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5 (1957), pp. 679–684. ISSN: 00959057, 19435274. URL: http://www.jstor.org/stable/24900506.

[12] Dimitri Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Vol. 27. Jan. 1996. DOI: 10.1007/978-0-387-74759-0_440.

[13] Justin Boyan and Andrew Moore. "Generalization in Reinforcement Learning: Safely Approximating the Value Function". In: *Advances in Neural Information Processing Systems*. Ed. by G. Tesauro, D. Touretzky, and T. Leen. Vol. 7. MIT

Press, 1995, pp. 369–376. URL: https://proceedings.neurips.cc/paper/1994/file/ef50c335cca9f340bde656363ebd02fd-Paper.pdf.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. USA: Cambridge University Press, 2004. ISBN: 0521833787.

[15] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[16] Yichen Chen and Mengdi Wang. *Stochastic Primal-Dual Methods and Sample Complexity of Reinforcement Learning*. 2016. arXiv: 1612.02516 [stat.ML].

[17] George Bernard Dantzig. *Linear Programming and Extensions*. Santa Monica, CA: RAND Corporation, 1963. DOI: 10.7249/R366.

[18] Peter Dayan. "Improving Generalization for Temporal Difference Learning: The Successor Representation". In: *Neural Computation* 5.4 (1993), pp. 613–624. DOI: 10.1162/neco.1993.5.4.613. eprint: https://doi.org/10.1162/neco.1993.5.4.613. URL: https://doi.org/10.1162/neco.1993.5.4.613.

[19] Peter Dayan. "The Convergence of TD(λ) for General λ". In: *Machine Learning* 8.3 (May 1992), pp. 341–362. DOI: 10.1023/A:1022632907294. URL: https://doi.org/10.1023/A:1022632907294.

[20] Daniela P. De Farias and Benjamin Van Roy. "On Constraint Sampling in the Linear Programming Approach to Approximate Dynamic Programming". In: *Mathematics of Operations Research - MOR* 29 (Aug. 2004), pp. 462–478. DOI: 10.1287/moor.1040.0094.

[21] Daniela P. De Farias and Benjamin Van Roy. "The linear programming approach to approximate dynamic programming". In: *Operations Research* 51 (2001), p. 2003.

[22] E.V. Denardo. *Dynamic Programming: Models and Applications*. Prentice-Hall, 1982. ISBN: 9780132215077. URL: https://books.google.ca/books?id=WSUoAQAAMAAJ.

[23] Eric V. Denardo. "On Linear Programming in a Markov Decision Problem". In: *Management Science* 16.5 (1970), pp. 281–288. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2628518.

[24] Marguerite Frank and Philip Wolfe. "An algorithm for quadratic programming". In: *Naval Research Logistics Quarterly* 3.1-2 (1956), pp. 95–110. DOI: 10.1002/nav.3800030109. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800030109. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800030109.

[25] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. "A Theory of Regularized Markov Decision Processes". In: ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 2160–2169. URL: http://proceedings.mlr.press/v97/geist19a.html.

[26] Guy T. de Ghellinck and Gary D. Eppen. "Linear Programming Solutions for Separable Markovian Decision Problems". In: *Management Science* 13.5 (1967), pp. 371–394. DOI: 10.1287/mnsc.13.5.371. eprint: https://doi.org/10.1287/mnsc.13.5.371. URL: https://doi.org/10.1287/mnsc.13.5.371.

[27] Ronald A. Howard. *Dynamic Programming and Markov Processes*. The MIT press, Cambridge, Massachusetts, 1960.

[28] Andrew Ilyas et al. "A Closer Look at Deep Policy Gradients". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=ryxdEkHtPS.

[29] Tommi Jaakkola, Michael Jordan, and Satinder Singh. "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms". In: *Neural Computation* 6 (Nov. 1994), pp. 1185–1201. DOI: 10.1162/neco.1994.6.6.1185.

[30] Nan Jiang et al. "The Dependence of Effective Planning Horizon on Model Accuracy". In: *AAMAS*. 2015.

[31] Sham Kakade and John Langford. "Approximately Optimal Approximate Reinforcement Learning". In: *19th International Conference on Machine Learning*. 2002, pp. 267–274.

[32] Sanket Kamthe and Marc Deisenroth. "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control". In: ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. Playa Blanca, Lanzarote, Canary Islands: PMLR, Sept. 2018, pp. 1701–1710. URL: http://proceedings.mlr.press/v84/kamthe18a.html.

[33] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Combinatorica* 4.4 (Dec. 1984), pp. 373–395. DOI: 10.1007/BF02579150. URL: https://doi.org/10.1007/BF02579150.

[34] Victor Klee and George J Minty. "How good is the simplex algorithm?" In: *Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)* (), pp. 159–175.

[35] Tejas D. Kulkarni et al. *Deep Successor Reinforcement Learning*. 2016. arXiv: 1606.02396 [stat.ML].

[36] L. G. Khachiyan. "A Polynomial-Time Algorithm for Solving Linear Programs". In: *Dokl. Akad. Nauk SSSR* 244 (5 1979), pp. 1093–1096. URL: http://mi.mathnet.ru/eng/dan42319.

[37] Michael Littman. "Algorithms for Sequential Decision Making". PhD thesis. Brown University, Aug. 2009.

[38] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. "On the Complexity of Solving Markov Decision Problems". In: UAI'95. Montréal, Québec, Canada: Morgan Kaufmann Publishers Inc., 1995, pp. 394–402. ISBN: 1558603859.

[39] Alan S. Manne. "Linear Programming and Sequential Decisions". In: *Management Science* 6.3 (1960), pp. 259–267. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2627340.

[40] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In: ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1928–1937. URL: http://proceedings.mlr.press/v48/mniha16.html.

[41] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 1476-4687. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236.

[42] Ofir Nachum and Bo Dai. *Reinforcement Learning via Fenchel-Rockafellar Duality*. 2020. arXiv: 2001.01866 [cs.LG].

[43] Ofir Nachum et al. *AlgaeDICE: Policy Gradient from Arbitrary Experience*. 2019. arXiv: 1912.02074 [cs.LG].

[44] Ofir Nachum et al. "DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 2318–2328. URL: https://proceedings.neurips.cc/paper/2019/file/cf9a242b70f45317ffd281241fa66502-Paper.pdf.

[45] Yurii Nesterov. "Primal-dual subgradient methods for convex problems". In: *Mathematical Programming* 120.1 (Aug. 2009), pp. 221–259. DOI: 10.1007/s10107-007-0149-x. URL: https://doi.org/10.1007/s10107-007-0149-x.

[46] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994. DOI: 10.1137/1.9781611970791. eprint: https://epubs.siam.org/doi/pdf/

10.1137/1.9781611970791. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611970791.

[47] Gergely Neu, Anders Jonsson, and Vicenç Gómez. *A unified view of entropy-regularized Markov decision processes*. 2017. arXiv: 1705.07798 [cs.LG].

[48] Chris Nota and Philip S. Thomas. "Is the Policy Gradient a Gradient?" In: *AAMAS*. 2020.

[49] OpenAI et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. arXiv: 1912.06680 [cs.LG].

[50] Dirk Ormoneit and Śaunak Sen. "Kernel-Based Reinforcement Learning". In: *Machine Learning* 49.2 (Nov. 2002), pp. 161–178. ISSN: 1573-0565. DOI: 10.1023/A:1017928328829. URL: https://doi.org/10.1023/A:1017928328829.

[51] Jason Pazis and Ronald Parr. "Non-Parametric Approximate Linear Programming for MDPs". In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI'11. San Francisco, California: AAAI Press, 2011, pp. 459–464.

[52] Ian Post and Yinyu Ye. "The simplex method is strongly polynomial for deterministic Markov decision processes". In: *CoRR* abs/1208.5083 (2012). arXiv: 1208.5083. URL: http://arxiv.org/abs/1208.5083.

[53] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1996.

[54] Balaraman Ravindran and Andrew Barto. "Approximate Homomorphisms : A framework for non-exact minimization in Markov Decision Processes". In: *Proceedings of the 5th International Conference on Knowledge-Based Computer Systems*. 2004.

[55] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *Ann. Math. Statist.* 22.3 (Sept. 1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

[56] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2005. ISBN: 0387212396.

[57] R. Tyrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970. ISBN: 9780691015866. URL: http://www.jstor.org/stable/j.ctt14bs1ff.

[58] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*. CUED/F-INFENG/TR 166. Cambridge University Engineering Department, Sept. 1994. URL: ftp://svr-ftp.eng.cam.ac.uk/reports/rummery_tr166.ps.Z.

[59] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597.

[60] Julian Schrittwieser et al. *Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model*. 2019. arXiv: 1911.08265 [cs.LG].

[61] John Schulman et al. "Trust Region Policy Optimization". In: ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1889–1897. URL: http://proceedings.mlr.press/v37/schulman15.html.

[62] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. ISSN: 1476-4687. DOI: 10.1038/nature24270. URL: https://doi.org/10.1038/nature24270.

[63] Richard S. Sutton. "Learning to predict by the methods of temporal differences". In: *Machine Learning* 3.1 (Aug. 1988), pp. 9–44. DOI: 10.1007/BF00115009. URL: https://doi.org/10.1007/BF00115009.

[64] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[65] Richard S Sutton et al. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 2000, pp. 1057–1063. URL: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[66] Philip Thomas. "Bias in Natural Actor-Critic Algorithms". In: ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, June 2014, pp. 441–448. URL: http://proceedings.mlr.press/v32/thomas14.html.

[67] John N. Tsitsiklis and Benjamin van Roy. "Feature-based methods for large scale dynamic programming". In: *Machine Learning* 22.1 (Mar. 1996), pp. 59–94. ISSN: 1573-0565. DOI: 10.1007/BF00114724. URL: https://doi.org/10.1007/BF00114724.

[68] John N. Tsitsiklis and Benjamin Van Roy. "Feature-Based Methods for Large Scale Dynamic Programming". In: *Machine Learning* 22.1 (Jan. 1996), pp. 59–94. DOI: 10.1023/A:1018008221616. URL: https://doi.org/10.1023/A:1018008221616.

[69] L. G. Valiant. "A Theory of the Learnable". In: *Commun. ACM* 27.11 (Nov. 1984), pp. 1134–1142. ISSN: 0001-0782. DOI: 10.1145/1968.1972. URL: https://doi.org/10.1145/1968.1972.

[70] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. *On Connections between Constrained Optimization and Reinforcement Learning*. 2019. arXiv: 1910.08476 [cs.LG].

[71] John. Von Neumann and A. H. Taub. *Collected works of J. von Neumann : general editor, A.H. Taub*. English. Oxford: Pergamon Press, 1961.

[72] T. Wang, M. Bowling, and D. Schuurmans. "Dual Representations for Dynamic Programming and Reinforcement Learning". In: *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. 2007, pp. 44–51.

[73] Christopher John Cornish Hellaby Watkins. "Learning from Delayed Rewards". PhD thesis. Cambridge, UK: King's College, May 1989. URL: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

[74] Lin Xiao. "Dual Averaging Method for Regularized Stochastic Learning and Online Optimization". In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio et al. Curran Associates, Inc., 2009, pp. 2116–2124. URL: http://papers.nips.cc/paper/3882-dual-averaging-method-for-regularized-stochastic-learning-and-online-optimization.pdf.

[75] Yinyu Ye. "The Simplex and Policy-Iteration Methods Are Strongly Polynomial for the Markov Decision Problem with a Fixed Discount Rate." In: *Mathematics of Operations Research* 36.4 (2011), pp. 593–603.