2016

Algorithms for First-order Sparse Reinforcement Learning

Bo Liu

University of Massachusetts - Amherst, boliu@cs.umass.edu

Follow this and additional works at: http://scholarworks.umass.edu/dissertations_2

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation
http://scholarworks.umass.edu/dissertations_2/588

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations May 2014 - current by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
ALGORITHMS FOR FIRST-ORDER SPARSE REINFORCEMENT LEARNING

A Dissertation Presented

by

BO LIU

Submitted to the Graduate School of the University of Massachusetts Amherst in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2016

College of Information and Computer Sciences
© Copyright by Bo Liu 2016
All Rights Reserved
ALGORITHMS FOR FIRST-ORDER SPARSE REINFORCEMENT LEARNING

A Dissertation Presented

by

BO LIU

Approved as to style and content by:

_____________________________
Sridhar Mahadevan, Chair

_____________________________
Andrew G. Barto, Member

_____________________________
Shlomo Zilberstein, Member

_____________________________
Weibo Gong, Member

_____________________________
James Allan, Chair
College of Information and Computer Sciences
ACKNOWLEDGMENTS

I would like to express my sincere thanks to my thesis advisor, Sridhar Mahadevan. Sridhar has been such a helpful advisor, and every aspect of this thesis has benefited from his guidance and support throughout my graduate studies. I also like to thank Sridhar for giving me great patience and support to explore many different ideas and research topics, and great trust when I had a hard time in publishing results and taking courses. I also appreciate the support offered by my other thesis committee members, Andrew Barto, Shlomo Zilberstein, and Weibo Gong.

I also thank my family, including my wife, Tingting Wang, and my baby boy, Nicolas Liu. I am also indebted to my parents, my parents-in-law, for their constant generous support. Love from the family is the most selfless shelter to me, and is also the resource where the sense of responsibility and motivation come from.

I am grateful for many other professors and staff members, who helped me along. I also thank Gwyn Mitchell, Susan Overstreet, Leanne Leclerc, and Barbara Sutherland for their help with my questions over the years. I thank the members of the Autonomous Learning Lab for the support and many helpful discussions. Thanks to the current lab members, Thomas Boucher, CJ Carey, Stefan Dernbach, Ishan Durugkar, Kristina Fedorenko Francisco M. Garcia, Ian Gemp, Stephen Giguere, Nicholas Jacek, Clemens Rosenbaum, Kevin Spliteri, Chris Vigorito. A lot of previous lab folks, including Philip Thomas, Thomas Helmuth, Bruno Castro da Silva,
William Dabney, Scott Niekum, Yariv Z. Levy, Scott Kuindersma, George Konidaris, Jeffrey Johns, Chang Wang, Sarah Osentoski, Khashayar Rohanimanesh, Mohammad Ghavamzadeh, and Georgios Theocharous, also rendered me great help in many aspects. Special thanks to Chang Wang, Ji Liu, Mohammad Ghavamzadeh, and Ian Gemp.

I have been fortunate to have great collaborators from industry in the past few years. I am grateful to the members of Amazon Transaction Risk Management team, and eBay Search Science. Thanks especially to: Yi Sun, Yangho Chen, Chongshan Zhang, Sunil Mohan.

I also like to thank my former advisors, teachers, and collaborators. Thanks to my advisors and collaborators in Stevens Institute of Technology, including Haibo He, Yudong Yao, Shuai Li, Sheng Chen, Hui Xiong, etc. I am also grateful to my advisors and collaborators in USTC, China, including Yong Wang, Baoqun Yin, Yan Xiong.
ABSTRACT

ALGORITHMS FOR FIRST-ORDER SPARSE REINFORCEMENT LEARNING

FEBRUARY 2016

BO LIU

B.Sc., NANJING UNIVERSITY OF POSTS & TELE., CHINA
M.Sc., UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA, CHINA
M.Sc., STEVENS INSTITUTE OF TECHNOLOGY
Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Sridhar Mahadevan

This thesis presents a general framework for first-order temporal difference learning algorithms with an in-depth theoretical analysis.

The main contribution of the thesis is the development and design of a family of first-order regularized temporal-difference (TD) algorithms using stochastic approximation and stochastic optimization. To scale up TD algorithms to large-scale problems, we use first-order optimization to explore regularized TD methods using linear value
function approximation. Previous regularized TD methods often use matrix inversion, which requires cubic time and quadratic memory complexity. We propose two algorithms, sparse-Q and RO-TD, for on-policy and off-policy learning, respectively. These two algorithms exhibit linear computational complexity per-step, and their asymptotic convergence guarantee and error bound analysis are given using stochastic optimization and stochastic approximation.

The second major contribution of the thesis is the establishment of a unified general framework for stochastic-gradient-based temporal-difference learning algorithms that use proximal gradient methods. The primal-dual saddle-point formulation is introduced, and state-of-the-art stochastic gradient solvers, such as mirror descent and extragradient are used to design several novel RL algorithms. Theoretical analysis is given, including regularization, acceleration analysis and finite-sample analysis, along with detailed empirical experiments to demonstrate the effectiveness of the proposed algorithms.
CONTENTS

ACKNOWLEDGMENTS ................................................................. iv

ABSTRACT .................................................................................. vi

LIST OF TABLES ........................................................................ xiv

LIST OF FIGURES ................................................................. xv

CHAPTER

1. INTRODUCTION ........................................................................ 1

1.1 Related Work ............................................................................ 2

1.2 Thesis Contributions ............................................................... 4

1.3 Summary of the Remaining Chapters ........................................ 6

2. BACKGROUND ........................................................................... 7

2.1 Reinforcement Learning .......................................................... 7

2.1.1 Basics of MDP and RL ....................................................... 7

2.1.2 Mathematical Formulations of Policy Evaluation Algorithms .... 9

2.1.2.1 Stochastic Variational Inequality Formulation of TD ........ 10
2.1.2.2 Matrix Inversion Formulation of LSTD ................. 14
2.1.2.3 Gradient-Motivated Algorithms ...................... 14
2.1.2.4 Summary ........................................ 16

2.1.3 Biased Sampling Problem .................................. 16

2.2 First-Order Optimization ..................................... 19

2.2.1 Proximal Point Methods ................................... 19

2.2.1.1 Stochastic Composite Optimization Formulation ..... 19
2.2.1.2 Proximal Gradient Method and Mirror Descent ...... 20

2.2.2 Operator Splitting ....................................... 23

2.2.2.1 Proximal Splitting: Coupled Objective Functions ... 23
2.2.2.2 Operator Splitting: Coupled Constraints .......... 25

2.3 Primal-Dual Splitting: Compound Operator Splitting .......... 26

2.3.1 Primal-Dual Saddle-Point Formulation ...................... 26

2.3.1.1 Basics of Saddle-Point Problems .................... 26
2.3.1.2 Primal-Dual Splitting and Convex Conjugate
          Functions ..................................... 27
2.3.1.3 Dual-norm Formulation ............................ 29

2.3.2 Primal-Dual Algorithm: An Overview .................... 30

2.3.2.1 Primal-Dual Algorithm ............................ 31
2.3.2.2 Extragradient .................................... 31
2.3.2.3 Mirror-Prox Algorithm ............................ 33

2.4 Summary .................................................. 34
3. SPARSE Q-LEARNING ......................................................... 36

3.1 Problem Formulation ....................................................... 37
3.2 Algorithm Design ............................................................ 39

3.2.1 TD Learning with Mirror Descent ................................ 39
3.2.2 Sparse Temporal Difference Learning with Mirror Descent . 40
3.2.3 Mirror Descent TD with AdaGrad .................................. 41
3.2.4 Choice of Distance-Generating Function ......................... 44

3.2.4.1 Vanilla Gradient and Regular TD Algorithm ................. 44
3.2.4.2 Multiplicative Gradient and $p$-norm TD Algorithm .......... 44
3.2.4.3 Exponentiated Gradient and Exponentiated TD Algorithm .. 45
3.2.4.4 Discussions ...................................................... 46

3.3 Theoretical Analysis .......................................................... 47
3.4 Empirical Experiments ........................................................ 48

3.4.1 Discrete MDPs ............................................................ 48
3.4.2 Continuous MDPs ......................................................... 49

3.5 Summary ................................................................. 50

4. REGULARIZED OFF-POLICY TD LEARNING ......................... 56

4.1 Introduction ................................................................. 56

4.1.1 Off-Policy Reinforcement Learning ............................... 56
4.1.2 Convex-concave Saddle-point First-order Algorithms ......... 57

4.2 Problem Formulation .......................................................... 59
4.2.1 Linear Inverse Problem Formulation .........................59
4.2.2 Un-squared Loss Formulation ..................................61
4.2.3 Squared Loss Formulation ..................................63

4.3 Algorithm Design ................................................64
4.3.1 RO-TD Algorithm Design ..................................64
4.3.2 RO-GQ(\(\lambda\)) Design ..................................66

4.4 Theoretical Analysis .............................................67
4.5 Empirical Results ................................................67
4.5.1 MSPBE Minimization and Off-Policy Convergence .............68
4.5.2 Feature Selection .............................................70
4.5.3 High-dimensional Under-actuated Systems ....................71

4.6 Summary ..................................................................72

5. FINITE-SAMPLE ANALYSIS OF PROXIMAL GRADIENT
   TD ALGORITHMS ............................................. 74

5.1 Introduction ....................................................75
5.2 Preliminaries ....................................................78
5.2.1 Gradient-based TD Algorithms ..............................78
5.2.2 Related Work ..............................................82

5.3 Saddle-point Formulation Of GTD Algorithms .................84
5.4 Finite-sample Analysis ...........................................86
5.4.1 The Revised GTD Algorithms ..............................87
5.4.2 Assumptions .................................................87
5.4.3 Finite-Sample Performance Bounds .......................90
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Mathematical Foundations of Policy Evaluation Algorithms</td>
</tr>
<tr>
<td>3.1</td>
<td>A Comparison of Different Mirror Descent Formulations</td>
</tr>
<tr>
<td>3.2</td>
<td>Results on Triple-Link Inverted Pendulum Task</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of TD, LARS-TD, RO-TD, $l_2$ LSTD, TDC and TD</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison of RO-GQ($\lambda$), GQ($\lambda$), and LARS-TD on Triple-Link Inverted Pendulum Task</td>
</tr>
<tr>
<td>5.1</td>
<td>Steady State Performance Comparison of Battery Management Domain</td>
</tr>
<tr>
<td>5.2</td>
<td>Steady State Performance Comparison of Bicycle Domain</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Contribution of The Thesis</td>
<td>6</td>
</tr>
<tr>
<td>2.1 The mirror descent method. This figure is adapted from Bubeck [2014]</td>
<td>24</td>
</tr>
<tr>
<td>2.2 The extragradient method</td>
<td>32</td>
</tr>
<tr>
<td>2.3 The Mirror-Prox method. This figure is adapted from Bubeck [2014]</td>
<td>35</td>
</tr>
<tr>
<td>3.1 Mirror-descent Q-learning converges significantly faster than LARS-TD on a “two-room” grid world MDP for $\gamma = 0.9$ (top left) and $\gamma = 0.8$ (top right). The y-axis measures the $l_2$ (red curve) and $l_\infty$ (blue curve) norm difference between successive weights during policy iteration. Bottom: running times for LARS-TD (blue solid) and mirror-descent Q (red dashed). Regularization $\rho = 0.01$.</td>
<td>51</td>
</tr>
<tr>
<td>3.2 Top figure: convergence of mirror-descent Q-learning with a fixed $p$-norm link function. Bottom figure: decaying $p$-norm link function.</td>
<td>52</td>
</tr>
<tr>
<td>3.3 Top: Convergence of AdaGrad Mirror-descent Q-learning on two-room gridworld domain. Bottom: Approximated value function, using 50 proto-value function bases.</td>
<td>53</td>
</tr>
</tbody>
</table>
3.4 Top: Q-learning; Bottom: mirror-descent Q-learning with $p$-norm link function, both with 25 fixed Fourier bases [Konidaris et al., 2011] for the mountain car task. ........................................54

3.5 Mirror-descent Q-learning on the Acrobot task using automatically generated diffusion wavelet bases averaged over 5 trials. .............55

4.1 Notation and Definitions ........................................61

4.2 Illustrative examples of the convergence of RO-TD using the Star and Random-walk MDPs. ........................................69

5.1 Off-Policy Convergence Comparison .........................104

5.2 Chain Domain ..................................................105

5.3 Energy Management Example .................................106

5.4 Energy Management Example .................................108

5.5 Summary of Comparisons between TD and GTD algorithm family......................................................109

A.1 Error Bound and Decomposition ..............................117
A fundamental challenge in artificial intelligence (AI) is to develop a computational framework for autonomous sequential decision-making through interactions with complex stochastic environments. One of the most widely studied frameworks for autonomous sequential decision-making is reinforcement learning (RL) [Sutton and Barto, 1998] that maximizes long-term reward in which an agent learns a policy which maps states to actions. RL models the interaction between the agent and an environment as a Markov Decision Process (MDP) [Puterman, 1994], and RL algorithms can be viewed as sample-based variants of classical methods for solving MDPs, such as policy iteration and value iteration. Temporal difference (TD) learning methods, which are widely used in solving RL problems [Sutton and Barto, 1998], have the ability to learn incrementally from samples of state transitions and rewards with complexity linear in the number of samples and state space representation, without requiring a model.

To ensure the learning algorithms tractable, a reasonable computational complexity is desired. This is especially important in domains whose state variables have many values, or that are high-dimensional, that is, that have many state variables. An algorithm should have a reasonable computational complexity that scales gracefully...
with the number of dimensions of real-world problems. With the help of modern stochastic optimization theories and tools, all the algorithms in the thesis have linear computational complexity with respect to both memory and the number of samples.

The major contribution of this thesis is a principled framework for finite-sample analysis of a new family of proximal gradient TD algorithms with linear (in both memory and time) computational complexity per-step. To the best of our knowledge, this thesis gives the first finite-sample analysis of linear TD algorithms. The thesis also explores a unified framework for designing stochastic gradient-based TD algorithms along with a detailed theoretical analysis of both asymptotic convergence and convergence rate analysis.

Regularization is known to play an important role in scaling up machine learning algorithms [Hastie et al., 2001]. Motivated by this, we explore regularization, especially $\ell_1$ sparsification of RL algorithms. Incorporating $\ell_1$ regularization in RL may help enhance model simplicity and generalization capability which will further help induce robust representations. This dissertation reports our design of a novel family of algorithms with the aforementioned properties, along with rigorous theoretical analysis including error-bound analysis and asymptotic convergence guarantees. We also evaluate the performance of the developed algorithms on various benchmark testbeds.

\section{Related Work}

First-order algorithms refer to algorithms that only make use of first-order information of the objective function, including the value of the objective function, and the
gradient/subgradient of the objective function. It also refers to algorithms with linear computational complexity per step with respect to the number of samples and the number of features of the problem. In this thesis, the term “first-order” refers to both: the algorithms only make use of the gradient/subgradient information, and also have linear computational complexity with respect to the sample size and the feature size.

TD algorithms form a widely-used class of first-order policy evaluation algorithms for RL problems. TD learning uses bootstrapping, i.e., the estimation of the value of the current state depends on the estimation of the value of the next state. Although TD converges when samples are drawn “on-policy” by sampling from the Markov chain underlying a policy in an MDP, it can be shown to be divergent when samples are drawn “off-policy.” Off-policy methods are more applicable since they can be used to learn while executing an exploratory policy, learn from demonstrations, and learn multiple tasks in parallel [Degris et al., 2012]. Sutton et al. [Sutton et al., 2008, 2009] introduced off-policy convergent algorithms, whose computation time per-step scales linearly with the number of samples and the number of features.

Regularizing RL algorithms lead to more robust methods that can scale up to large problems with many potentially irrelevant features. Regularized Policy Iteration [Farahmand et al., 2008] is an example of $\ell_2$-regularized approaches in kernelized reinforcement learning. Kolter and Ng [2009] introduced the LARS-TD approach, by combining $\ell_1$ regularization using Least Angle Regression (LARS) [Efron et al., 2004] with the LSTD framework. Another approach was introduced in [Johns et al., 2010], termed LCP-TD, based on the Linear Complementary Problem (LCP) formulation, a variational inequality (VI) approach. LCP-TD uses “warm-starts”, which helps to significantly reduce the burden of $\ell_1$ regularization. A theoretical analysis of $\ell_1$ reg-
ularization was given in [Ghavamzadeh et al., 2011], including error bound analysis with finite samples in the on-policy setting. Another approach [Geist et al., 2012] integrates the Dantzig Selector [Candes and Tao, 2007] with LSTD, and thus overcomes some of the drawbacks of LARS-TD. An approximate linear programming approach for finding $\ell_1$ regularized solutions of the Bellman equation was presented in [Petrik et al., 2010]. All of these approaches require per-step complexity approximately cubic in the number of (active) features, and thus only work well for moderate-sized problems.

1.2 Thesis Contributions

The major contribution of this thesis is a new framework for first-order RL algorithms leveraged by the advances in first-order stochastic optimization.

- We propose a framework for designing first-order RL algorithms with stochastic optimization. One long-standing problem with current first-order RL algorithms is that very few first-order TD algorithms are true stochastic gradient methods with respect to an objective function. Barnard [1993] proved that the TD learning is not a true stochastic gradient method by showing that the TD update cannot be derived as the gradient of any function [Barnard, 1993]; Although the GTD family of algorithms [Maei, 2011] are derived from the gradient of an objective function such as MSPBE and NEU, they have not been proven to be true stochastic gradient methods in the sense that the expected weight update direction can be different from the direction descending the gradient of the objective function for which they were designed [Szepesvári, 2010]. Residual gradient (RG) [Baird, 1995] method, which min-
imizes the mean-square TD error, might be the only gradient-based TD algo-

rithm thus far. However, both empirical results [Dann et al., 2014] and theo-

retical analysis [Sutton et al., 2009] show that the solution of the RG method

is inferior to other solutions. Therefore, there is a large gap between stochastic

optimization and first-order RL algorithms, which makes the complexity anal-

ysis, incremental regularization and convergence rate acceleration difficult for

first-order RL algorithms.

In this thesis, our goal is to formulate RL algorithms using a stochastic op-

timization framework. We introduce convex-concave saddle-point formulation

into first-order RL and present a novel framework for regularized off-policy RL

algorithms. More specifically, we use the saddle-point representation to design

regularized RL algorithms, and we propose the first off-policy convergent TD al-

gorithm with linear computational complexity. The merit of this work is that it

provides a general primal-dual framework that allows a variety of regularization

schemes (not limited to $\ell_1$ sparsification) with provable results on convergence

rate and error bound.

• We introduce several stochastic optimization tools to help accelerate first-order

RL algorithms and help achieve (near)-optimal convergence rate. Such opti-

mization tools include mirror descent and extragradient acceleration. Experi-

mental results show that these tools are powerful for RL algorithms in reducing

variance, accelerating convergence rate and increasing the robustness of the

algorithms.

The contributions of the thesis are illustrated in Figure 1.1.
1.3 Summary of the Remaining Chapters

In Chapter 2, we provide background knowledge of RL and stochastic optimization. In Chapter 3, we propose a dual space sparse TD algorithm and the control learning extension, SparseQ. In Chapter 4, we introduce the saddle-point framework into RL and propose the RO-TD algorithm. In Chapter 5, we present finite-sample analysis of TD learning with linear computational complexity (per-step), and propose accelerated algorithms based on the primal-dual saddle-point formulation.
CHAPTER 2
BACKGROUND

In this chapter, we introduce the basics of RL, including the Markov decision process (MDP) model, and value function approximation. For stochastic optimization, we first introduce two key ingredients, the proximal point method and primal-dual operator splitting, and then some algorithms, such as mirror descent and extragradient methods.

2.1 Reinforcement Learning

2.1.1 Basics of MDP and RL

RL [Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998] is a class of algorithms for finding good approximations to a class of learning problems in which an agent interacts with an unfamiliar, dynamic and stochastic environment with the goal of optimizing some measure of its long-term performance. This interaction is conventionally modeled as a Markov decision process (MDP). A MDP is defined as the tuple \((\mathcal{S}, \mathcal{A}, P, R, \gamma)\), where \(\mathcal{S}\) and \(\mathcal{A}\) are finite sets of states and actions, the transition kernel \(P(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]\) specifies the probability of transition from state \(s \in \mathcal{S}\) to state \(s' \in \mathcal{S}\) by taking action \(a \in \mathcal{A}\), \(R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}\) is the reward function bounded by \(R_{\text{max}}\), and \(0 \leq \gamma < 1\) is a discount factor. A stationary policy
π : S × A → [0, 1] is a mapping from states to actions. The main objective of an RL algorithm is to find an good approximation to the optimal policy. In order to achieve this goal, a key step in many algorithms is to calculate the value function of a given policy π, i.e., Vπ : S → R, a process known as policy evaluation. It is known that Vπ is the unique fixed-point of the Bellman operator Tπ, i.e.,

\[ Vπ = TπVπ = Rπ + γPπVπ, \] (2.1)

where Rπ and Pπ are the reward function and transition kernel of the Markov chain induced by policy π. In Eq. (2.1), we may imagine Vπ as an |S|-dimensional vector and write everything in vector/matrix form. In the following, to simplify the notation, we often drop the dependence of Tπ, Vπ, Rπ, and Pπ on π.

Let πb denote the behavior policy that generates the data, and let π denote the target policy that we want to evaluate. They are the same in the on-policy setting and different in the off-policy setting. For each state-action pair (si, ai), such that πb(ai|si) > 0, we define the importance-weighting factor ρi = π(ai|si)/πb(ai|si) with ρmax ≥ 0 being its maximum value over the state-action pairs.

When S is too large to use the tabular representation method, we often use a linear approximation architecture for Vπ with parameters θ ∈ Rd and L-bounded basis functions \{ϕi\}d i=1, i.e., ϕi : S → R and maxi ||ϕi||∞ ≤ L. We denote by φ(⋅) = (ϕ1(⋅), ..., ϕd(⋅))T the feature vector and by F the linear function space spanned by the basis functions \{ϕi\}d i=1, i.e., F = \{fθ | θ ∈ Rd and fθ(⋅) = φ(⋅)Tθ\}.

We may write the approximation of V in F in the vector form as ̂v = Φθ, where Φ is the |S| × d feature matrix. When only n training samples of the form D =
\[
\{(s_i, a_i, r_i = r(s_i, a_i), s'_i)\}_{i=1}^n, \quad s_i \sim \xi, \quad a_i \sim \pi_b(\cdot|s_i), \quad s'_i \sim P(\cdot|s_i, a_i),
\]
are available (\(\xi\) is a distribution over the state space \(S\)). We may write the empirical Bellman operator \(\hat{T}\) for a function in \(F\) as
\[
\hat{T}(\hat{\Phi}\theta) = \hat{R} + \gamma \hat{\Phi}'\theta,
\]
where \(\hat{\Phi}\) (respectively \(\hat{\Phi}'\)) is the empirical feature matrix of size \(n \times d\), whose \(i\)-th row is the feature vector \(\phi(s_i)\) (respectively \(\phi(s'_i)\)), and \(\hat{R} \in \mathbb{R}^n\) is the reward vector, whose \(i\)-th element is \(r_i\). We denote by \(\delta_i(\theta) = r_i + \gamma \phi_i'\theta - \phi_i\theta\), the TD error for the \(i\)-th sample \((s_i, r_i, s'_i)\) and define \(\Delta \phi_i = \phi_i - \gamma \phi'_i\). Finally, we define the matrices \(A\) and \(C\), and the vector \(b\) as
\[
A := \mathbb{E}[\phi_i(\Delta \phi_i)\top], \quad b := \mathbb{E}[\phi_i r_i], \quad C := \mathbb{E}[\phi_i \phi_i\top],
\]
where the expectations are with respect to \(\xi\) and \(P^{a_i}\). We also denote by \(\Xi\), the diagonal matrix whose elements are \(\xi(s)\), and \(\xi_{\text{max}} := \max_s \xi(s)\). For each sample \(i\) in the training set \(D\), we can calculate unbiased estimates of \(A\), \(b\), and \(C\) as follows:
\[
\hat{A}_i := \phi_i \Delta \phi_i\top, \quad \hat{b}_i := r_i \phi_i, \quad \hat{C}_i := \phi_i \phi_i\top.
\]

### 2.1.2 Mathematical Formulations of Policy Evaluation Algorithms

In this section, we give a comprehensive overview of the mathematical formulations of existing policy evaluation algorithms. It should be mentioned that in contrast to the empirical comparison of existing surveys such as Dann et al. [2014], this section does not provide any empirical comparison but tries to set up the mathematical framework of existing policy evaluation algorithms.
2.1.2.1 Stochastic Variational Inequality Formulation of TD

The first type of policy evaluation algorithm is the *stochastic variational inequality* family of algorithms. A common characteristic of this family of algorithms is that at each step, the weights are updated in a recursive way. A lot of policy evaluation methods can be categorized as recursive least-squares methods [Scherrer and Geist, 2012], such as Fixed-point Kalman Filter (FKPF) [Choi and Van Roy, 2006], Least-squares Policy Evaluation (LSPE) [Bertsekas and Ioffe, 1996], etc. To illustrate this framework, let us first review TD learning. Here, we give two perspectives of the TD algorithm. The first perspective interprets TD algorithm as a stochastic approximation approach, i.e., the stochastic approximation of the solution of a linear inverse problem $A\theta = b$, where $A, b$ are defined in Eq. (2.3).

The TD algorithm, in essence, solves the linear equation $A\theta = b$ using a stochastic approximation approach as

\begin{align*}
\theta_0 &= 0, \\
\theta_{t+1} &= \theta_{t+1} + \alpha_t (\hat{b}_t - \hat{A}_t \theta_t),
\end{align*}

(2.5) (2.6)

$\hat{A}_t, \hat{b}_t$ are the stochastic unbiased estimation of $A, b$ respectively, which are defined in Eq. (2.4). Although $\hat{A}_t$ is a square matrix, and the computation looks like $O(n^2)$ per-step, since $\hat{A}_t$ is a rank-1 matrix, using matrix computation association rule, the computational cost is linear per-step, which is computationally attractive.

This is a very straightforward perspective, and is widely recognized. However, an obvious drawback of this perspective is that this formulation does not easily allow regularization. Furthermore, it does not reveal the intrinsic relation between TD al-
gorithm and other recursive least-squares based algorithms, such as FPKF and LSPE. Hence, we shed another perspective, which formulates TD algorithm as a stochastic variational inequality (SVI) problem, which is motivated by Bertsekas [2011].

The (deterministic) variational inequality (VI) problem formulation of the linear equation $A\theta = b$ is formulated as

$$\langle A\theta^* - b, \theta - \theta^* \rangle \geq 0 \quad (2.7)$$

This can be solved by a recursive proximal mapping

$$\theta_{t+1} = \arg \min_x \left\{ \langle x, A\theta_t - b \rangle + \frac{1}{2\alpha_t} ||x - \theta_t||_2^2 \right\} \quad (2.8)$$

However, in real applications, $A, b$ are not accessible. If we replace $A, b$ with $\hat{A}_t, \hat{b}_t$ defined in Eq. (2.4), and define the TD error for the $t$-th sample as

$$\delta_t(\theta) := r_i + \gamma \phi_t^\top \theta_t - \phi_i^\top \theta_t. \quad (2.9)$$

It is easy to verify that $\delta_t(\theta) = \hat{b}_t - \hat{A}_t \theta_t$. Then we have the SVI problem formulation of TD as solving a recursive proximal mapping at each step, i.e.,

$$\theta_{t+1} = \arg \min_x \left\{ \langle x, \hat{A}_t \theta_t - \hat{b}_t \rangle + \frac{1}{2\alpha_t} ||x - \theta_t||_2^2 \right\} \quad (2.10)$$

$$= \arg \min_x \left\{ \langle x, -\phi_t \delta_t \rangle + \frac{1}{2\alpha_t} ||x - \theta_t||_2^2 \right\} \quad (2.11)$$

So at each iteration, the update law is

$$\theta_{t+1} = \theta_t + \alpha_t \phi_t \delta_t \quad (2.12)$$
which is the update law of the TD algorithm. Note that in the TD algorithm, at each iteration,

1. The quadratic term (termed as prox-function) in the recursive proximal mapping in Eq. (2.11) is currently used as $\frac{1}{2}||x - \theta_t||_2^2$, which does not take into consideration the sample distribution $\Xi$. It would be preferable to use the proximal function as

$$\frac{1}{2}||\Phi x - \Phi \theta_t||_\Xi^2 = \frac{1}{2}||x - \theta_t||_C^2,$$

(2.13)

where $C := \Phi^\top \Xi \Phi = \mathbb{E}[\phi_i \phi_i^\top]$. So if we use the prox-function as defined in Eq. (2.13), we have the recursive proximal update as

$$\theta_{t+1} = \arg\min_x \left\{ \langle x, A_t \theta_t - b_t \rangle + \frac{1}{2\alpha_t} ||x - \theta_t||_C^2 \right\}$$

(2.14)

The update law is written as

$$\theta_{t+1} = \theta_t + \alpha_t C^{-1}(b_t - A_t \theta_t)$$

(2.15)

This requires computing $C^{-1}$, which is usually $O(d^3)$ and hence, computationally expensive. Using Sherman-Morris-Woodbury theorem, $C^{-1}$ can be computed iteratively with $O(d^2)$ computational complexity per-step.\(^1\) The update

\(^1\)According to Sherman-Morris-Woodbury theorem, given a square nonsingular matrix $A$, there is

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}$$

(2.16)
law is

\[
Z_{t+1} = Z_t - \frac{Z_t \phi_t \phi_t^\top Z_t}{1 + \phi_t^\top Z_t \phi_t} \tag{2.17}
\]

\[
\theta_{t+1} = \theta_t + \alpha_t Z_t \phi_t \delta_t, \tag{2.18}
\]

where \(Z_t\) is used to approximate \(C^{-1}\).

2. The stochastic unbiased estimation \(A_t, b_t\) is based on the current example. The estimation based on a single sample often suffers from high variance. If we can use the information of all the samples along the trajectory, then the estimation of \(A, b\) will be of lower variance. i.e., a more sample efficient approach is to store a trajectory-based estimation of the \(A, b\) matrix as

\[
\bar{A}_t = \frac{1}{t} \sum_{i=1}^{t} \hat{A}_i = \frac{1}{t} \sum_{i=1}^{t} \phi_i (\phi_i - \gamma \phi_i')^\top, \tag{2.19}
\]

\[
\bar{b}_t = \frac{1}{t} \sum_{i=1}^{t} \hat{b}_i = \frac{1}{t} \sum_{i=1}^{t} \phi_i r_i \tag{2.20}
\]

If we use all the samples along the whole trajectory, we obtain the recursive proximal mapping as follows,

\[
\theta_{t+1} = \arg \min_x \left\{ \langle x, \bar{A}_t \theta_t - \bar{b}_t \rangle + \frac{1}{2 \alpha_t} \|x - \theta_t\|_C^2 \right\}, \tag{2.21}
\]

where \(\bar{A}_t, \bar{b}_t\) is defined as Eq. (2.20). Thus the update law is

\[
Z_{t+1} = Z_t - \frac{Z_t \phi_t \phi_t^\top Z_t}{1 + \phi_t^\top Z_t \phi_t} \tag{2.22}
\]

\[
\theta_{t+1} = \theta_t + \alpha_t Z_t (\bar{b}_t - \bar{A}_t \theta_t) \tag{2.23}
\]
It should be mentioned that with these changes, the SVI problem formulation of FPKF and LSPE also change correspondingly. The VI formulation of FPKF and LSPE can be written as

\[ \langle C^{-1}(A\theta^* - b), \theta - \theta^* \rangle \geq 0, \]  
\[ (2.24) \]

The aforementioned stochastic variational inequality formulation requires that the Bellman operator \( T \) should be a contraction mapping.

2.1.2.2 Matrix Inversion Formulation of LSTD

The LSTD solution is using matrix inversion to solve the linear equation \( A\theta = b \) since \( A \) is a full-rank square matrix. The solution is \( \theta = A^{-1}b \), and the computational cost for computing \( A^{-1} \) is \( O(d^2) \) by using the Sherman-Morris-Woodbury theorem. It is obvious that this algorithm requires that \( A \) is an invertible, linear operator (i.e. invertible matrix), which requires two conditions: 1) the Bellman operator \( T \) is a contraction mapping, and 2) a linear value function architecture should be used.

2.1.2.3 Gradient-Motivated Algorithms

The above mentioned stochastic approximation and matrix inversion methods are widely used, especially TD algorithm and LSTD algorithm, which are considered the most successful policy evaluation algorithms with linear \( O(d) \) and quadratic \( O(d^2) \) computational complexity, respectively. However, these two methods have some intrinsic drawbacks as follows.
1. Off-policy convergence is not guaranteed. As can be seen from the above analysis, both the SVI formulation and the matrix inversion formulation requires that $A$ is invertible. This in turn requires that the Bellman operator is a contraction mapping, which only holds for on-policy settings. It is desirable if an optimization-based algorithm could be designed.

2. Regularization is very difficult to implement. Since these aforementioned algorithms are based on solving the linear inverse problem $A\theta = b$, the regularization term is not easy to add (except in the special case of ridge regression, $||\theta||^2_2$).

3. It’s difficult to obtain theoretical convergence rates for stochastic approximation and matrix inversion approaches, not to mention finite-sample analyses. Recently, the online-learning finite sample analysis of LSTD was performed via treating LSTD as a Markov regression model [Lazaric et al., 2010b]. There is also an analysis based on the dual form of GTD2 algorithm [Valcarcel Macua et al., 2015]. However, such analysis only holds for on-policy settings. To the best of our knowledge, there have not been any convergence rate and finite-sample analysis results for off-policy policy evaluation algorithms; Also, there is no finite-sample analysis for policy evaluation algorithms with linear computation complexity.

[Sutton et al., 2009] proposed the GTD family algorithms, aiming to set up a stochastic optimization framework that has off-policy convergence guarantees. Motivated by the fact that the TD/LSTD solution converges to the fixed-point equation of $V_{\theta} = \Pi TV_{\theta}$, a new objective function termed the mean-square projected Bellman-error (MSPBE) is defined as
Another widely used objective function is called the norm of the expected TD update (NEU), which does not seem to have an obvious geometric interpretation, however, the TD solution finds its minimum. The NEU objective function is defined as

$$\text{NEU}(\theta) = \mathbb{E}[\delta_t(\theta)\phi_t] \mathbb{E}^\top[\delta_t(\theta)\phi_t].$$

2.1.2.4 Summary

We summarize the mathematical formulations and restrictions of the above existing policy evaluation algorithms in Table 2.1.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Essence</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD/FPKF/LSPE</td>
<td>stochastic variational inequality</td>
<td>$T$ should be a contraction mapping</td>
</tr>
<tr>
<td>LSTD</td>
<td>matrix inversion solution of linear equation</td>
<td>Linear VFA, $T$ should be a contraction</td>
</tr>
<tr>
<td>GTD/GTD2</td>
<td>optimization</td>
<td>$T$ should be differentiable with respect to $\theta$</td>
</tr>
</tbody>
</table>

Table 2.1. Mathematical Foundations of Policy Evaluation Algorithms

$$\text{MSPBE}(\theta) = \|V_\theta - \Pi TV_\theta\|^2_\Xi.$$  \hspace{1cm} (2.25)
deterministic or the model of the environment is known, then two independent samples can be generated from the model, which is not always the case.

Biased sampling is a long-standing problem that has troubled RL researchers for decades. Biased sampling is often caused by the product of the Bellman errors, or the product of the Bellman error and the gradient of Bellman error. Due to the stochastic nature of the MDP, given a state-action pair \((s_t, a_t)\), there may be many successive state \(s'_t\), thus these products cannot be consistently estimated via a single sample. This problem hinders the objective functions to be solved via stochastic gradient descent (SGD) algorithms. \textit{Unfortunately, any Bellman error based objective function suffers from the biased sampling problem.} Consider, for example, the NEU objective function in Equation (2.26). Taking the gradient with respect to \(\theta\), we have

\[
-\frac{1}{2} \nabla \text{NEU}(\theta) = \mathbb{E}[(\phi_t - \gamma \phi'_{t}) \phi_t^\top] \mathbb{E}[\delta_t(\theta) \phi_t]
\]  

(2.27)

If the gradient can be written as a single expectation value, then it is straightforward to use a stochastic gradient method, however, here we have a \textit{product of correlated expectations}, and due to the correlation between \((\phi_t - \gamma \phi'_{t}) \phi_t^\top\) and \(\delta_t(\theta) \phi_t\), the sampled product is not an unbiased estimate of the gradient. In other words, \(\mathbb{E}[(\phi_t - \gamma \phi'_{t}) \phi_t^\top]\) and \(\mathbb{E}[\delta_t(\theta) \phi_t]\) can be directly sampled, yet \(\mathbb{E}[(\phi_t - \gamma \phi'_{t}) \phi_t^\top] \mathbb{E}[\delta_t(\theta) \phi_t]\) cannot be directly sampled.

A generalization of the reason is stated as follows. Consider any Bellman error based objective function \(J(\theta)\), which can be represented as

\[
J(\theta) = f(g(\theta, s, s')), \quad \text{(2.28)}
\]
where \( g(\theta, s, s') \) is an expectation function, and \( f \) is a mapping on a vector field to a scalar, such as a norm operator. The generalization form of the gradient \( \nabla J(\theta, s, s') \) is computed as

\[
\nabla_{\theta}J(\theta) = \nabla_{\theta}f(g(\theta, s, s'))\nabla_{\theta}g(\theta, s, s').
\] (2.29)

We can see that due to the product of \( \nabla_{\theta}f(g(\theta, s, s')) \) and \( \nabla_{\theta}g(\theta, s, s') \), biased sampling is unavoidable. In sum, any Bellman error based objective function will cause biased-sampling problem for direct stochastic gradient method. Based on the above analysis, it is obvious that the existing objective functions cannot be solved via stochastic gradient approach due to the biased sampling problem.

There are several attempts to address the biased sampling problem. The first attempt is the RG algorithm [Baird, 1995], which aims to solve the MSBE objective function, yet due to the aforementioned biased sampling problem, RG does not minimize the MSBE objective function, but instead the Mean-Square TD-error (MSTDE) objective, as proven in [Maei, 2011]. Recently, attempts have been made to minimize the MSPBE/NEU objectives with a two-time-scale technique to address the biased sampling problem by Sutton et al. [2008, 2009]. However, the two-time-scale technique made the method not a true stochastic gradient method in the sense that the expected weights update direction can be different from the direction of the true negative gradient of the objective function.
2.2 First-Order Optimization

2.2.1 Proximal Point Methods

In this section, we introduce two optimization techniques, i.e., proximal point method and operator splitting, which constitute the cornerstone of many modern optimization algorithms.

2.2.1.1 Stochastic Composite Optimization Formulation

Stochastic optimization explores the use of first-order gradient methods for solving convex optimization problems. We first give some definitions before moving on to introduce stochastic composite optimization.

Definition 1. (Lipschitz-continuous Gradient): The gradient of a closed convex function \( f(x) \) is \( L \)-Lipschitz continuous if \( \exists L, \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \forall x, y \in X \).

Definition 2. (Strong Convexity): A convex function is \( \mu \)-strongly convex if \( \exists \mu, \mu^2\|x - y\|^2 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle, \forall x, y \in X \).

Remark: If \( f(x) \) is both \( L \)-Lipschitz continuous gradient and \( \mu \)-strongly convex, then \( \forall x, y \in X, \)

\[
\frac{\mu}{2}\|x - y\|^2 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2}\|x - y\|^2
\]

Definition 3. (Stochastic Subgradient) : The stochastic subgradient for closed convex function \( f(x) \) at \( x \) is defined as \( g(x, \xi) \) satisfying \( \mathbb{E}[g(x, \xi)] := \nabla f(x) \in \partial f(x) \), where
\( \partial f(x) \) is the subgradient of \( f \) with respect to \( x \). Further, we assume that the variance is bounded \( \exists \sigma > 0 \) such that

\[
\forall x \in X, \mathbb{E}[\|g(x, \xi_t) - \nabla f(x)\|^2_2] \leq \sigma^2
\] (2.30)

Here we define the problem of Stochastic Composite Optimization (SCO)[Lan, 2012]:

**Definition 4.** *(Stochastic Composite Optimization):* A stochastic composite optimization problem \( F(L, M, \mu, \sigma) : \Psi(x) \) on a closed convex set \( X \) is defined as

\[
\min_{x \in X} \Psi(x) \overset{\text{def}}{=} f(x) + h(x),
\] (2.31)

where \( f(x) \) is a convex function with \( L \)-Lipschitz continuous gradient and \( h(x) \) is a convex Lipschitz continuous function such that

\[
|h(x) - h(y)| \leq M\|x - y\|, \forall x, y \in X,
\] (2.32)

where \( g(x, \xi_t) \) is the stochastic subgradient of \( \Psi(x) \) defined above with variance bound \( \sigma \). Such \( \Psi(x) \) is termed as a \( F(L, M, \mu, \sigma) \) problem.

### 2.2.1.2 Proximal Gradient Method and Mirror Descent

Before we move on to introduce mirror descent, we first give some definitions and notation.

**Definition 5.** *(Distance-generating Function)*[Beck and Teboulle, 2003]: A distance-generating function \( \psi(x) \) is a continuously differentiable \( \mu \)-strongly convex function. \( \psi^* \) is the Legendre transform of \( \psi \), which is defined as \( \psi^*(y) = \sup_{x \in X} (\langle x, y \rangle - \psi(x)) \).
Definition 6. (Bregman Divergence)/[Beck and Teboulle, 2003]: Given distance-generating function $\psi$, the Bregman divergence induced by $\psi$ is defined as:

$$ D_\psi(x, y) = \psi(x) - \psi(y) - \langle \nabla \psi(y), x - y \rangle $$  \hspace{1cm} (2.33) 

The Legendre transform and Bregman divergence have the following properties:

- $\nabla \psi^* = (\nabla \psi)^{-1}$
- $\nabla D_\psi(u, v) = \nabla \psi(u) - \nabla \psi(v)$

An interesting choice of the link function $\psi(\cdot)$ is the ($q - 1$)-strongly convex function $\psi(\theta) = \frac{1}{2} \|\theta\|_q^2$, and $\psi^*(\tilde{\theta}) = \frac{1}{2} \|\tilde{\theta}\|_p^2$. Here, $\|\theta\|_q = \left( \sum_j |\theta_j|^q \right)^{\frac{1}{q}}$, and $p$ and $q$ are conjugate numbers such that $\frac{1}{p} + \frac{1}{q} = 1$ [Gentile, 2003]. $\theta$ and $\tilde{\theta}$ are conjugate variables in primal space and dual space, respectively.

$$ \nabla \psi(\theta)_j = \frac{\text{sign}(\theta_j) |\theta_j|^{q-1}}{\|\theta\|_q^{q-2}} $$

$$ \nabla \psi^*(\tilde{\theta})_j = \frac{\text{sign}(\tilde{\theta}_j) |\tilde{\theta}_j|^{p-1}}{\|\tilde{\theta}\|_p^{p-2}} $$  \hspace{1cm} (2.34)

Also it is worth noting that when $p = q = 2$, the Legendre transform is the identity mapping.

We now introduce the concept of proximal mapping, and then describe the mirror descent framework. The proximal mapping associated with a convex function $h(x)$ is defined as:

$$ \text{prox}_h(x) = \arg\min_{u \in \mathcal{X}} (h(u) + \frac{1}{2} \|u - x\|^2) $$  \hspace{1cm} (2.35)
In the case of \( h(x) = \rho \|x\|_1 (\rho > 0) \), which is particularly important for sparse feature selection, the proximal operator turns out to be the soft-thresholding operator \( S_\rho(\cdot) \), which is an entry-wise shrinkage operator that moves a point towards zero, i.e.,

\[
prox_h(x)_i = S_\rho(x)_i = \text{sign}(x_i) \max(|x_i - \rho|, 0)
\] (2.36)

where \( i \) is the index, and \( \rho \) is a threshold. With this background, we now introduce the proximal gradient method. At each iteration, the optimization sub-problem of Equation (2.31) can be rewritten as

\[
x_{t+1} = \arg \min_{u \in X} \left( h(u) + \langle \nabla f_t, u \rangle + \frac{1}{2\alpha_t} \|u - x_t\|^2 \right)
\] (2.37)

If computing \( prox_h \) is not expensive, then computation of Equation (2.31) is of the following formulation, which is called the proximal gradient method

\[
x_{t+1} = prox_{\alpha_t h}(x_t - \alpha_t \nabla f(x_t))
\] (2.38)

where \( \alpha_t > 0 \) is stepsize, constant or determined by line search. The mirror descent [Beck and Teboulle, 2003] algorithm is a generalization of classic gradient descent, which has led to developments of new more powerful machine learning methods for classification and regression. Mirror descent can be viewed as an enhanced gradient method, particularly suited to minimization of convex functions in high-dimensional spaces. Unlike traditional gradient methods, mirror descent undertakes gradient updates of weights in the dual space, which is linked together with the primal space using a Legendre transform. Mirror descent can be viewed as a proximal algorithm in
which the distance-generating function is a Bregman divergence with respect to the
distance-generating function $\psi$, and thus the optimization problem is

$$
\text{prox}_h(x) = \arg \min_{u \in X} (h(u) + D_\psi(u, x))
$$

(2.39)

The solution to this optimization problem of Equation (2.39) forms the core procedure of *mirror descent* as a generalization of Equation (2.37)

$$
x_{t+1} = \arg \min_{u \in X} \left( h(u) + \langle \nabla f_t, u \rangle + \frac{1}{\alpha_t} D_\psi(u, x_t) \right)
$$

(2.40)

which is a nonlinear extension of Equation (2.38)

$$
x_{t+1} = \nabla \psi^*(\text{prox}_{\alpha_t h} (\nabla \psi(x_t) - \alpha_t \nabla f(x_t)))
$$

(2.41)

Mirror descent has become the cornerstone of many online $\ell_1$ regularization approaches such as in [Shalev-Shwartz and Tewari, 2011], Xiao [2010] and Duchi et al. [2010]. Figure 2.1 illustrates the mirror descent method.

### 2.2.2 Operator Splitting

In this section, we introduce operator splitting, which is another significant achievement in modern optimization theory.

#### 2.2.2.1 Proximal Splitting: Coupled Objective Functions

In this section we give a brief overview of proximal splitting algorithms [Combettes and Pesquet, 2011]. The two key ingredients of proximal splitting are proximal oper-
Figure 2.1. The mirror descent method. This figure is adapted from Bubeck [2014].

ators and operator splitting. Operator splitting is widely used to reduce the computational complexity of many optimization problems, resulting in algorithms such as sequential non-iterative approach (SNIA), Strang splitting, and sequential iterative approach (SIA). Proximal splitting is a technique that combines proximal operators and operator splitting, and deals with problems where the proximal operator is difficult to compute at first, yet is easier to compute after decomposition. The very basic scenario is Forward-Backward Splitting (FOBOS) [Duchi and Singer, 2009]

\[
\min_{\theta \in \mathcal{X}} (\Psi(\theta) = f(\theta) + h(\theta)) \quad (2.42)
\]

where \( f(\cdot) \) is a convex, continuously differentiable function with \( L \)-Lipschitz-continuous bounded gradients, i.e. \( \forall x, y, \| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \| \), and \( h(\theta) \) is a convex (possibly not smooth) function. FOBOS solves this problem via the following proximal gradient method

\[
\theta_{t+1} = \text{prox}_{\alpha h}(\theta_t - \alpha_t \nabla f(\theta_t)) \quad (2.43)
\]

24
2.2.2.2 Operator Splitting: Coupled Constraints

Another widely used operator splitting technique is called alternating direction method of multipliers (ADMM) [Eckstein and Bertsekas, 1992], which solves convex optimization problems with coupled equality constraints such as $Ax + By = C$. The problem formulation is as

$$\min_{x,y} f(x) + g(y) \quad \text{s.t.} \quad Ax + By = C \quad (2.44)$$

In such formulation, the objective function is separable with respect to different sets of variables, and the equality constraints is the coupled equality with respect to these sets of variables. The Augmented Lagrangian Multiplier (ALM) with respect to Eq. (2.44) is

$$L_\rho(x, y, \lambda) = f(x) + g(y) + \lambda^\top (Ax + By - C) + \frac{\rho}{2} ||Ax + By - C||^2 \quad (2.45)$$

At each iteration, the ADMM solver deals with one variable at a time as follows,

$$x_{t+1} = \arg \min_x L_\rho(x_t, y_t, \lambda)$$
$$y_{t+1} = \arg \min_y L_\rho(x, y_t, \lambda)$$
$$\lambda_{t+1} = \lambda_t + \rho(Ax_{t+1} + By_{t+1} - C) \quad (2.46)$$

Such methods enables parallel computation, and thus form the cornerstones of many existing large-scale optimization algorithms, as summarized in [Boyd et al., 2011].
2.3 Primal-Dual Splitting: Compound Operator Splitting

In the above section, we introduced two popular splitting methods, i.e., FOBOS for splitting a sum of objective functions, and ADMM for splitting equality constraints. However, FOBOS and ADMM both address splitting the sum of objective functions/constraints. Sometimes the problem is formulated as $\min_{x \in X} f(g(x))$, where $f(\cdot)$ is a lower semi-continuous (l.s.c) nonlinear convex function, and $g$ is a linear operator. In the following, we denote $f(g(x))$ as $f \circ g(x)$. In some problems, $\text{prox}_f(\cdot)$ and $\text{prox}_g(\cdot)$ are easy to compute, yet $\text{prox}_{f \circ g}(\cdot)$ is difficult to compute. An intuitive idea is to find an operator-splitting technique that is able to split the compound operator $f \circ g(x)$. We introduce primal-dual splitting, which serves the purpose of splitting the compound operator $f \circ g(x)$.

2.3.1 Primal-Dual Saddle-Point Formulation

Motivated by the introduction in the previous section, we introduce primal-dual splitting, which constitutes the major theoretical foundation of the proximal gradient TD learning framework.

2.3.1.1 Basics of Saddle-Point Problems

In this section, we introduce some basics of convex-concave saddle-point problems, which is a type of well-studied non-convex problem. A convex-concave saddle-point problem is formulated as follows. Let $x \in X, y \in Y$, where $X, Y$ are both nonempty bounded closed convex sets, and $f(x) : X \to \mathbb{R}$ be a convex function. If there exists a function $L(\cdot, \cdot)$ such that $f(x)$ can be represented as $f(x) := \sup_{y \in Y} L(x, y)$, then the optimization problem of minimizing $f$ over $X$ is converted into an equivalent convex-
concave saddle-point problem \( \min_{x \in X} \max_{y \in Y} L(x, y) \). There are some basic properties of saddle-point problems \( L(x, y) \) which will be frequently used.

**Lemma 1. (Saddle-Point Inequality)**[Boyd and Vandenberghe, 2004] For saddle-point problem \( L(x, y) \), we have the following inequality

\[
\min_x L(x, y) \leq \max_y \min_x L(x, y) \leq \min_x \max_y L(x, y) \leq \max_y L(x, y)
\]  

(2.47)

**Definition 7. (Error Function)** The error function of the saddle-point problem (5.9) at each point \((\theta', y')\) is defined as

\[
\text{Err}(\theta', y') = \max_y L(\theta', y) - \min_{\theta} L(\theta, y').
\]  

(2.48)

### 2.3.1.2 Primal-Dual Splitting and Convex Conjugate Functions

In this section, we introduce a compound operator splitting mechanism termed primal-dual splitting. The problem formulation we tackle is

\[
\min_{x \in X} (\Psi(x) = F(Ax - b) + h(x))
\]  

(2.49)

where \( X, Y \) are convex compact sets, \( Ax - b \) is a vector-valued affine operator, where \( A : X \to Y \) is a matrix, and \( b \in Y \) is a vector, \( F : Y \to \mathbb{R} \) is a lower-semicontinuous (l.s.c) nonlinear convex function. The proximal operator with respect to this problem is \( \text{prox}_{F(Ax - b)} \). In many cases, although \( \text{prox}_F \) and \( \text{prox}_{Ax - b} \) are easy to compute, \( \text{prox}_{F(Ax - b)} \) is often difficult to compute. To address this problem, we use the primal-dual splitting framework to facilitate operator splitting, i.e., which only uses \( \text{prox}_F, \text{prox}_{Ax - b} \), and avoids computing \( \text{prox}_{F(Ax - b)} \) directly.
Before moving on, we first introduce convex conjugate functions. The convex conjugate function is defined as follows

**Definition 8.** The conjugate of a function $f$ is defined as

$$f^*(y) = \sup_x (y^\top x - f(x))$$

(2.50)

The conjugate function has some nice properties as introduced in [Boyd and Vandenberghe, 2004]. First, $f^*$ is convex and closed even if $f$ is not. Second, from the definition of $f^*$, it is easy to deduce the following

**Lemma 2. (Fenchel inequality)**

$$\forall x, y, f(x) + f^*(y) \geq y^\top x$$

(2.51)

Next, from the definition of $f^*$, it is easy to have the definition of $f^{**}$ as

$$f^{**}(x) = \sup_y (y^\top x - f^*(y))$$

(2.52)

Next we prove a useful lemma.

**Lemma 3.** $(f^{**})$ We have $f^{**} \leq f$; if $f$ is closed and convex, then $f^{**} = f$.

*Proof.** The proof can be found in Boyd and Vandenberghe [2004].

With these background, we can have
Lemma 4. if $f$ is closed and convex, then

$$f(x) = \sup_y (y^\top x - f^*(y))$$

(2.53)

Proof. The proof is explicit by the fact that if $f$ is closed and convex, we have

$$f(x) = f^{**}(x) = \sup_y (y^\top x - f^*(y))$$

(2.54)

This completes the proof.

With Lemma 4, the corresponding primal-dual formulation of Equation (2.49) can be deduced as

$$\min_{x \in X} \max_{y \in Y} (L(x, y) = \langle Ax - b, y \rangle - F^*(y) + h(x))$$

(2.55)

where $F^*(y) := \sup_{x \in X} (\langle x, y \rangle - F(x))$ is the convex conjugate of $F(\cdot)$. Particularly, a special $f$ worth discussing is $f(x) = \frac{1}{2} \|Ax - b\|_{C^{-1}}^2$, where $C$ is a positive definite (PD) symmetric matrix. It can be inferred that

$$f(x) = \frac{1}{2} \|Ax - b\|_{C^{-1}}^2 = \max_y \langle Ax - b, y \rangle - \frac{1}{2} \|y\|_{C}^2,$$

(2.56)

which will be frequently used in the rest of the thesis.

2.3.1.3 Dual-norm Formulation

Besides the general framework of the primal-dual formulation, there are several primal-dual splitting framework with specific $f$. One example is the dual norm representation.
Definition 9. Given a norm $\| \cdot \|$, the dual norm is defined as

$$
\| x \|_* = \max_y x^T y, \quad s.t. \quad \| y \| \leq 1
$$

(2.57)

As an example, consider $f(x) = \| Ax - b \|_m$ which admits a bilinear minimax representation

$$
f(x) := \| Ax - b \|_m = \max_{\| y \|_n \leq 1} (\langle y, Ax - b \rangle)
$$

(2.58)

where $m, n$ are conjugate numbers such that $\frac{1}{m} + \frac{1}{n} = 1$. Using the approach in [Nemirovski et al., 2009], Equation (2.58) can be solved as

$$
x_{t+1} = x_t - \alpha_t \langle y_t, A \rangle, \ y_{t+1} = \Pi_{\| y \|_n \leq 1}(y_t + \alpha_t(Ax_t - b))
$$

(2.59)

where $\Pi_{\| y \|_n \leq 1}$ is the projection operator of $y_t$ onto the unit-$l_n$ ball $\| y \|_n \leq 1$, which is defined as

$$
\Pi_{\| y \|_n \leq 1} y = \min(1, 1/\| y \|_n) y, \ n = 2, \ (\Pi_{\| y \|_n \leq 1} y)_i = \min(1, 1/|y_i|) y_i, \ n = \infty \quad (2.60)
$$

and $\Pi_{\| y \|_\infty \leq 1} y$ is an entry-wise operator.

2.3.2 Primal-Dual Algorithm: An Overview

In this section, we introduce saddle-point problem solvers. We introduce the primal-dual algorithm, extragradient algorithm, and Mirror-Prox algorithm. The saddle-point problem is formulated as $\min_{x \in X} \max_{y \in Y} L(x, y)$. 

30
Algorithm 1 Primal-Dual Algorithm Template

INPUT: saddle-point problem \( \min_x J(x) = \min_x \max_{y \in Y} L(\theta, y) + h(x) \), stepsize \( \{\alpha_t\} \)

OUTPUT: REPEAT

\[
x_{t+1} = \text{prox}_{x_t}(-\alpha_t \partial L_x(x_t, y_t)), \quad y_{t+1} = \text{prox}_{y_t}(\alpha_t \partial L_y(x_t, y_t))
\]  \hspace{1cm} (2.62)

UNTIL some stopping criteria is met

2.3.2.1 Primal-Dual Algorithm

The primal-dual algorithm is probably the most intuitive algorithm. For saddle-point problem \( \min_x \max_y L(x, y) \), at each iteration, we do gradient descent with respect to \( x \) and gradient ascent with respect to \( y \) (see Algorithm 1), which is in essence the classic Arrow-Hurwicz algorithm [Arrow et al., 1972]. At the \( t \)-th iteration, \( x_{t+1}, y_{t+1} \) are updated as

\[
x_{t+1} = \text{prox}_{x_t}(-\alpha_t \partial L_x(x_t, y_t)), \quad y_{t+1} = \text{prox}_{y_t}(\alpha_t \partial L_y(x_t, y_t))
\]  \hspace{1cm} (2.61)

However, there are more powerful saddle-point algorithms, as we show in the following sections.

2.3.2.2 Extragradient

The extragradient is first proposed by Korpelevich [1976] as a relaxation of gradient descent to solve variational inequality (VI) problems [Nagurney, 2013]. Conventional ordinary gradient descent can be used to solve VI problems only if some strict restrictions such as strong monotonicity of the operator or compactness of the feasible set are satisfied. The extragradient is proposed to solve VIs with relaxation on the aforementioned strict restrictions. The essence of extragradient methods is that instead
of moving along the steepest gradient descent direction with respect to the initial point in each iteration, two steps, i.e., an extrapolation step and a gradient descent step, are taken. In the extrapolation step, a step is made along the steepest gradient descent direction of the initial point, resulting in an intermediate point which is used to compute the gradient. Then the gradient descent step is made from the initial point in the direction of the gradient with respect to the intermediate point. The extragradient take steps as follows

\[
x_{t+\frac{1}{2}} = \Pi_X (x_t - \alpha_t \nabla f(x_t)) \\
x_{t+1} = \Pi_X (x_t - \alpha_t \nabla f(x_{t+\frac{1}{2}}))
\]

(2.63)

\(\Pi_X(x) = \arg \min_{y \in X} \|x - y\|^2\) is the projection onto the convex set \(X\), and \(\alpha_t\) is a stepsize. Convergence of the iterations of Equation (2.63) is guaranteed under the constraints \(0 < \alpha_t < \frac{1}{\sqrt{2L}}\) [Nemirovski, 2005], where \(L\) is the Lipschitz constant for \(\nabla f(x)\). Figure 2.2 illustrates the extragradient method.

![Figure 2.2. The extragradient method.](image-url)
Algorithm 2 Extragradient Algorithm Template

INPUT: saddle-point problem $\min_x J(x) = \min_x \max_{\theta, y} L(\theta, y) + h(x)$

OUTPUT: REPEAT

\[
x_{t+\frac{1}{2}} = \text{prox}_{x_t} (-\alpha_t \partial L_x(x_t, y_t)), \quad y_{t+\frac{1}{2}} = \text{prox}_{y_t} (\alpha_t \partial L_y(x_t, y_t)) \tag{2.66}
\]

\[
x_{t+1} = \text{prox}_{x_t} (-\alpha_t \partial L_x(x_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})), \quad y_{t+1} = \text{prox}_{y_t} (\alpha_t \partial L_y(x_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})) \tag{2.67}
\]

UNTIL some stopping criteria is met

For our saddle-point problem $\min_{x \in X} \max_{y \in Y} L(x, y)$, the update law is as follows

\[
x_{t+\frac{1}{2}} = \text{prox}_{x_t} (-\alpha_t \partial L_x(x_t, y_t)), \quad y_{t+\frac{1}{2}} = \text{prox}_{y_t} (\alpha_t \partial L_y(x_t, y_t)) \tag{2.64}
\]

\[
x_{t+1} = \text{prox}_{x_t} (-\alpha_t \partial L_x(x_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})), \quad y_{t+1} = \text{prox}_{y_t} (\alpha_t \partial L_y(x_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})) \tag{2.65}
\]

The general algorithm template of extragradient algorithm is

In Figure 2.2, the concept of extragradient is illustrated. A simple way to understand the figure is to imagine the vector field $F$ here is defined as the gradient $\nabla f(x)$ of some function being minimized. In that case, the mapping $-F(x_t)$ points as usual in the direction of the negative gradient. However, the clever feature of extragradient is that it moves not in the direction of the gradient at $x_k$, but rather in the direction of the negative gradient at the point $y_k$, which is the projection of the original gradient step onto the feasible set $K$.

2.3.2.3 Mirror-Prox Algorithm

Mirror-Prox algorithm, introduced by Nemirovski [2005], is an acceleration of the vanilla gradient algorithm that is widely used for large-scale problems [Juditsky and Nemirovski, 2011]. Later, stochastic Mirror-Prox algorithm (SMP) [Juditsky et al.,
Algorithm 3 Mirror-Prox Algorithm Template

**INPUT:** saddle-point problem \( \min_x J(x) = \min_x \max_{y \in Y} L(\theta, y) + h(x) \), distance-generating function \( \psi \)

**OUTPUT:** REPEAT

\[
[u_t, v_t] = \nabla \psi([x_t, y_t]) \quad (2.68)
\]

\[
u_{t+\frac{1}{2}} = \text{prox}_{u_t}(-\alpha_t \partial L_x(x_t, y_t)), \quad v_{t+\frac{1}{2}} = \text{prox}_{v_t}(\alpha_t \partial L_y(x_t, y_t)) \quad (2.69)
\]

\[
u_t+1 = \text{prox}_{u_t}(-\alpha_t \partial L_x(x_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})), \quad v_{t+1} = \text{prox}_{v_t}(\alpha_t \partial L_y(x_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})) \quad (2.70)
\]

\[
[x_{t+1}, y_{t+1}] = \nabla \psi^*([u_{t+1}, v_{t+1}]) \quad (2.71)
\]

UNTIL some stopping criteria is met

2008] is proposed to solve stochastic saddle-point problems (SSP). We refer the readers to [Yu et al., 2014] for a brief overview of MP algorithm and [Juditsky and Nemirovski, 2011] for a more details. The general algorithm template of Mirror-Prox algorithm is

The Mirror-Prox method generalizes the extragradient method to non-Euclidean geometries, analogous to the way mirror descent generalizes the regular gradient method. The Mirror-Prox algorithm (MP) [Nemirovski, 2005] is a first-order approach that is able to solve saddle-point problems at a convergence rate of \( O(1/t) \). Figure 2.3 illustrates the Mirror-Prox method.

### 2.4 Summary

In this section, we introduced the background knowledge of RL and MDP, which are the mathematical foundations of sequential decision-making. Then we introduced proximal point method and operator splitting, which lay the mathematical founda-
tions of the thesis. We also introduced several important techniques used in stochastic optimization, such as mirror descent and extragradient. All these techniques are widely used in optimization and comprise the cornerstones of the algorithms proposed in this thesis.

Figure 2.3. The Mirror-Prox method. This figure is adapted from Bubeck [2014].
CHAPTER 3

SPARSE Q-LEARNING

In this chapter we explore a new framework for (on-policy convergent) TD learning algorithm based on mirror descent and related algorithms. Mirror descent can be viewed as an enhanced gradient method, particularly suited to minimization of convex functions in high-dimensional spaces. Unlike traditional gradient methods, mirror descent undertakes gradient updates of weights in both the dual space and primal space, which are linked together using a Legendre transform. Mirror descent can be viewed as a proximal algorithm where the distance-generating function used is a Bregman divergence. A new class of proximal-gradient based temporal-difference (TD) methods are presented based on different Bregman divergences, which are more powerful than regular TD learning. Examples of Bregman divergences that are studied include $p$-norm functions, and Mahalanobis distance based on the covariance of sample gradients. A new family of sparse mirror-descent RL methods are proposed [Mahadevan and Liu, 2012], which are able to find the $\ell_1$-regularized value function for a fixed policy at significantly less computational cost than previous methods based on second-order matrix methods.
3.1 Problem Formulation

The problem formulation in this chapter is based on the Lasso-TD objective defined as follows, which is used in LARS-TD and LCP-TD. In this section, we extend the stochastic variational inequality (SVI) formulation proposed in Section 2 to incorporate sparsity. We then define $\ell_1$-regularized projection, and then give the definition of Lasso-TD objective function.

In the SVI formulation, we use sample-based estimation of $\hat{A}_t, \hat{b}_t$ defined in Eq. (2.4) to estimate the $A, b$ matrix, and the update law is the vanilla TD algorithm,

$$
\theta_{t+1} = \arg \min_x \left\{ \langle x, \hat{A}_t \theta_t - \hat{b}_t \rangle + \frac{1}{2\alpha_t} \|x - \theta_t\|_2^2 \right\}
$$

(3.1)

$$
= \arg \min_x \left\{ \langle x, -\phi_t \delta_t \rangle + \frac{1}{2\alpha_t} \|x - \theta_t\|_2^2 \right\}
$$

(3.2)

So at each iteration, the update law is

$$
\theta_{t+1} = \theta_t + \alpha_t \phi_t \delta_t
$$

(3.3)

which is the update law of the TD algorithm.

Next, if we consider incorporating regularization $h(\theta)$ (e.g., $h(\theta) = \rho \|\theta\|_1$ for sparsity, we would have the following SVI formulation

$$
\theta_{t+1} = \arg \min_x \left\{ \langle x, \hat{A}_t \theta_t - \hat{b}_t \rangle + \rho \|x\|_1 + \frac{1}{2\alpha_t} \|x - \theta_t\|_2^2 \right\}
$$

(3.4)

$$
= \arg \min_x \left\{ \langle x, -\phi_t \delta_t \rangle + \rho \|x\|_1 + \frac{1}{2\alpha_t} \|x - \theta_t\|_2^2 \right\}
$$

(3.5)
So at each iteration, the update law is

\[ \theta_{t+1} = S_{\alpha_t \rho} (\theta_t + \alpha_t \phi_t \delta_t), \]  

where the soft-thresholding operator is defined in Eq. (2.36).

Next we review some results from [Ghavamzadeh et al., 2011].

**Definition 10.** [Ghavamzadeh et al., 2011] \((\ell_1\text{-regularized Projection})\): \(\Pi_{\ell_1}\) is the \(\ell_1\)-regularized projection defined as:

\[ \Pi_{\ell_1} y = \Phi \theta, \theta = \arg\min_x \|y - \Phi x\|^2 + \rho \|x\|_1 \]  

(3.7)

It has been shown that \(\Pi_{\ell_1}\) is a non-expansive mapping with respect to weighted \(l_2\) norm, as proven in [Ghavamzadeh et al., 2011].

**Lemma 5.** [Ghavamzadeh et al., 2011]: \(\Pi_{\rho}\) is a non-expansive mapping such that

\[ \forall x, y \in R^d, \|\Pi_{\rho} x - \Pi_{\rho} y\|^2 \leq \|x - y\|^2 - \|x - y - (\Pi_{\rho} x - \Pi_{\rho} y)\|^2 \]  

(3.8)

**Definition 11.** [Ghavamzadeh et al., 2011] \((\text{Lasso-TD})\) Lasso-TD is a fixed-point equation with respect to \(\ell_1\) regularization with parameter \(\rho\), which is defined as

\[ \theta = f(\theta) = \arg\min_{x \in R^d} (\|T \Phi \theta - \Phi x\|^2 + \rho \|x\|_1) \]

\[ = \arg\min_{x \in R^d} (\|R^\pi + \gamma P^\pi \Phi \theta - \Phi x\|^2 + \rho \|x\|_1) \]  

(3.9)

The solution properties of Lasso-TD is discussed in detail in [Ghavamzadeh et al., 2011]. Note that the above \(\ell_1\) regularized fixed-point is not a convex optimization
problem but a fixed-point problem. Several prevailing sparse RL methods use Lasso-TD as the objective function, such as LARS-TD [Kolter and Ng, 2009] and LCP-TD [Johns et al., 2010]. The advantage of LARS-TD comes from LARS in that it computes a homotopy path of solutions with different regularization parameters, and thus offers a rich solution family. The major drawback comes from LARS, too. To maintain the LARS criteria in which each active variable has the same correlation with the residual, variables may be added and dropped several times, which is computationally expensive. In fact, the computational complexity per iteration is $O(ndk^2)$ where $k$ is the cardinality of the active feature set. When $k$ is comparable to $d$, the computational complexity deteriorate to $O(nd^3)$. Secondly, LARS-TD requires the $A$ matrix to be a $P$-matrix (a square matrix which does not necessarily to be symmetric, but all the principal minors are positive), which poses extra limitation on applications. LCP-TD [Johns et al., 2010] formulates LASSO-TD as a Linear Complementarity Problem (LCP), which can be solved by a variety of available LCP solvers.

3.2 Algorithm Design

3.2.1 TD Learning with Mirror Descent

Algorithm 4 describes the proposed mirror-descent TD($\lambda$) method. Unlike regular TD, the weights are updated using the TD error in the dual space by mapping the primal weights $\theta$ using a gradient of a strongly convex function $\psi$. Subsequently, the updated dual weights are converted back into the primal space using the gradient of the Legendre transform of $\psi$, namely $\nabla \psi^\star$. Algorithm 1 specifies the mirror descent

---

1All the algorithms described extend to the action-value case where $\phi(s)$ is replaced by $\phi(s, a)$.
TD(λ) algorithm in which each weight \( \theta_i \) is associated with an eligibility trace \( e(i) \).

For \( \lambda = 0 \), this is just the features of the current state \( \phi(s_t) \), but for nonzero \( \lambda \), this corresponds to a decayed set of features proportional to the recency of state visitations. Note that the distance-generating function \( \psi_t \) is a function of time.

**Algorithm 4 Mirror Descent TD(λ)**

Let \( \pi \) be some fixed policy for an MDP \( M \), and \( s_0 \) be the initial state. Let \( \Phi \) be some fixed or automatically generated basis, \( \psi \) is a distance-generating function.

1. **REPEAT**
   2. Do action \( \pi(s_t) \) and observe next state \( s_{t+1} \) and reward \( r_t \).
   3. Update the eligibility trace \( e_t \leftarrow \lambda \gamma e + \phi(s_t) \), \( \delta_t = r_t + \gamma \phi(s_{t+1})^\top \theta_t - \phi(s_t)^\top \theta_t \)
   4. Update the dual weights \( \tilde{\theta}_{t+1} \) for a linear function approximator:
      \[
      \tilde{\theta}_{t+1} = \nabla \psi_t(\theta_t) + \alpha_t \delta_t e_t
      \]
   5. Set \( \theta_{t+1} = \nabla \psi^*_t(\tilde{\theta}_{t+1}) \) where \( \psi^* \) is the Legendre transform of \( \psi \).
   6. Set \( t \leftarrow t + 1 \).
   7. **UNTIL** \( t = N \).

Return \( V_\theta \approx \Phi \theta_t \) as the value function associated with policy \( \pi \) for MDP \( M \).

### 3.2.2 Sparse Temporal Difference Learning with Mirror Descent

The proposed mirror-descent RL framework enables new first-order algorithms for learning sparse solutions value function representations that are more scalable than previous matrix-based sparse RL methods.

Algorithm 5 describes a modification to obtain sparse value functions resulting in a sparse mirror-descent TD(λ) algorithm. The main difference is that the dual weights \( \theta \) are truncated to satisfy the \( \ell_1 \) penalty on the weights. Here, \( \rho \) is the sparsity pa-
rameter defined in Equation (3.9). An analogous approach was suggested in [Shalev-Shwartz and Tewari, 2011] for $\ell_1$ penalized classification and regression.

**Algorithm 5** Sparse Mirror Descent TD($\lambda$)

Let $\pi$ be some fixed policy for an MDP $M$, and $s_0$ be the initial state. Let $\Phi$ be some fixed or automatically generated basis, $\psi$ is a distance-generating function.

1. **REPEAT**

2. Do action $\pi(s_t)$ and observe next state $s_{t+1}$ and reward $r_t$.

3. Update the eligibility trace $e_t \leftarrow \lambda e_t + \phi(s_t), \delta_t = r_t + \gamma \phi(s_{t+1})^\top \theta_t - \phi(s_t)^\top \theta_t$

4. Update the dual weights $\tilde{\theta}_{t+1}$ and truncate weights:

   $$\tilde{\theta}_{t+1} = S_{\alpha_t \rho}(\nabla \psi_t(\theta_t) + \alpha_t \delta_t e_t)$$

   (e.g., $\psi(\theta) = \frac{1}{2}\|\theta\|_q^2$ is the $p$-norm link function).

5. $\theta_{t+1} = \nabla \psi^*_t(\tilde{\theta}_{t+1})$ (e.g., $\psi^*(\theta) = \frac{1}{2}\|\theta\|_p^2$ and $p$ and $q$ are dual norms such that $\frac{1}{p} + \frac{1}{q} = 1$).

6. Set $t \leftarrow t + 1$.

7. **UNTIL** $t = N$.

Return $V_\theta \approx \Phi \theta_t$ as the $\ell_1$ penalized sparse value function associated with policy $\pi$ for MDP $M$.

### 3.2.3 Mirror Descent TD with AdaGrad

Another possible mirror-descent TD algorithm uses as the distance-generating function a Mahalanobis distance derived from the subgradients generated during actual trials. We base our derivation on the AdaGrad approach proposed in [Duchi et al., 2011] for classification and regression. Here we introduce some background knowledge. Given a positive definite matrix $A$, the Mahalanobis norm of a vector $x$ is defined as $\|x\|_A = \sqrt{x, Ax}$. Let $g_t = \partial f(s_t)$ be the subgradient of the function be-
ing minimized at time $t$, and $G_t = \sum_t g_t g_t^\top$ be the covariance matrix of outer products of the subgradients.

The AdaGrad [Duchi et al., 2011] is motivated by the fact that no all features are equally important, and thus the learning rate should be adjusted entry-wise with respect to each feature. The AdaGrad is a very clever trick that uses historical information to help the entry-wise learning rate automatically adapt to the information geometry of the learning problem (as revealed through the historical gradients): rather than the Euclidean distance, it uses the Mahalanobis distance with covariance $G_t$. It is computationally more efficient to use the diagonal matrix $H_t = \sqrt{\text{diag}(G_t)}$ instead of the full covariance matrix, which can be expensive to estimate. This amounts to use a time-dependent Bregman divergence in the proximal gradient formulation,

$$\theta_{t+1} = \arg\min_{x \in X} \left( \langle x, \partial f_t \rangle + h(x) + \frac{1}{\alpha_t} D_{\psi_t}(x, \theta_t) \right) \quad (3.10)$$

Here, $h$ serves as a fixed regularization function, such as the $\ell_1$ penalty, and $\psi_t$ is the time-dependent distance-generating function as in mirror descent, and the time-dependent Bregman divergence is $\psi_t = \frac{1}{2} || \cdot ||^2_{H_t}$. Thus the proximal gradient formulation is

$$\theta_{t+1} = \arg\min_{x \in X} \left( \langle x, \partial f_t \rangle + h(x) + \frac{1}{2\alpha_t} || x - \theta_t ||^2_{H_t} \right) \quad (3.11)$$

The update law is thus written as
Motivated by the success and simplicity of AdaGrad, we propose Algorithm 6, which describes the adaptive subgradient mirror descent TD method.

**Algorithm 6 AdaGrad Mirror Descent TD(λ)**

Let π be some fixed policy for an MDP $M$, and $s_0$ be the initial state. Let $Φ$ be some fixed or automatically generated basis, $ψ$ is a distance-generating function, $G_0 = 0$.

1. **REPEAT**

2. Do action $π(s_t)$ and observe next state $s_{t+1}$ and reward $r_t$.

3. Update the eligibility trace $e_t ← \lambda \gamma e_t + \phi(s_t), \delta_t = r_t + \gamma \phi(s_{t+1})^\top \theta_t - \phi(s_t)^\top \theta_t$.

4. Update feature covariance and Mahalanobis matrix $H_t$

   $$G_t = G_{t-1} + \phi(s_t)\phi(s_t)^\top, H_t = \sqrt{\text{diag}(G_t)}$$

5. Update the weights $\theta_{t+1}$ according to Eq. (3.13).

6. Set $t ← t + 1$.

7. **UNTIL** $t = N$.

Return $V_\theta ≈ Φ\theta_t$ as the $\ell_1$ penalized sparse value function associated with policy $π$ for MDP $M$.

**Remark:** As can be seen from the update law, if there is no sparsification step, then the update law reduces to

$$\theta_{t+1,i} = \theta_{t,i} + \frac{α_t \delta_t e_t}{H_{t,ii}}$$ (3.14)
3.2.4 Choice of Distance-Generating Function

We now discuss various choices for the distance-generating function in Algorithm 4. It should be noted that the topic of the choice of distance-generating function equals the choice of link function and that of the Bregman divergence function. We start from the simplest case of the vanilla gradient and regular TD learning, then expose the relation between different distance-generating functions, the corresponding gradient method, and the corresponding TD-based algorithms.

3.2.4.1 Vanilla Gradient and Regular TD Algorithm

In the simplest case, suppose $\psi(\theta) = \frac{1}{2}\|\theta\|_2^2$, the Euclidean length of $\theta$. In this case, it is easy to see that mirror descent TD($\lambda$) corresponds to regular TD($\lambda$), since the gradients $\nabla\psi$ and $\nabla\psi^*$ correspond to the identity function.

3.2.4.2 Multiplicative Gradient and $p$-norm TD Algorithm

A much more interesting choice of $\psi$ is $\psi(\theta) = \frac{1}{2}\|\theta\|_{\frac{q}{p}}^2$, and its conjugate Legendre transform $\psi^*(\tilde{\theta}) = \frac{1}{2}\|\tilde{\theta}\|_p^2$. Here, $\|\theta\|_q = \left(\sum_j |\theta_j|^q\right)^{\frac{1}{q}}$, and $p$ and $q$ are conjugate numbers such that $\frac{1}{p} + \frac{1}{q} = 1$. This $\psi(\theta), \psi^*(\tilde{\theta})$ leads to the $p$-norm link function $\tilde{\theta} = \nabla\psi(\theta), \theta = \nabla\psi^*(\tilde{\theta})$ which are mappings: $\mathbb{R}^d \rightarrow \mathbb{R}^d$ [Gentile, 2003]:

$$\tilde{\theta}_j = (\nabla\psi(\theta))_j = \frac{\text{sign}(\theta_j)|\theta_j|^{q-1}}{\|\theta\|_q^{q-2}}, \quad \theta_j = (\nabla\psi^*(\tilde{\theta}))_j = \frac{\text{sign}(\tilde{\theta}_j)|\tilde{\theta}_j|^{p-1}}{\|\tilde{\theta}\|_p^{p-2}}$$

(3.15)

The $p$-norm function has been extensively studied in the literature on online learning [Gentile, 2003], and it is well-known that for large $p$, the corresponding classification or regression method behaves like a multiplicative method (e.g., the $p$-norm regression
method for large $p$ behaves like an exponentiated gradient method (EG) [Kivinen and Warmuth, 1995; Littlestone, 1988]).

### 3.2.4.3 Exponentiated Gradient and Exponentiated TD Algorithm

Another distance-generating function is the negative entropy function $\psi(\theta) = \sum_i \theta_i \log \theta_i$, which leads to the entropic mirror descent algorithm [Beck and Teboulle, 2003]. Interestingly, this special case has been previously explored [Precup and Sutton, 1997] as the exponentiated-gradient TD method, although the connection to mirror descent and Bregman divergences were not made in this previous study, and EG does not generate sparse solutions [Shalev-Shwartz and Tewari, 2011]. The link function in exponentiated gradient is

$$\nabla \psi = \log, \quad \nabla \psi^* = (\log)^{-1}$$  \hspace{1cm} (3.16)

We can see that the sparse mirror descent algorithm can be divided into the following steps:

- Use the mirror-map $x_t$ to the dual space and obtain $\nabla \psi(x_t)$.
- Do gradient descent in the dual space and obtain $\bar{y}_t$.
- Do sparsification using soft-thresholding function and obtain $y_t$.
- Use the inverse mirror-map to map back to primal space and obtain $x_{t+1}$.

From the steps, we can see that since the sparsification step is done in the dual space, thus to preserve sparsity, the link function should be entrywise sparsity preserving.
Namely, if the $i$-th entry $y_{t,i} = 0$, then to preserve sparsity, then we should have $x_{t+1,i} = \nabla \psi(\tilde{y}_{t,i}) = 0$ as well. The identity link function and the $p$-norm link function both satisfies this property; Unfortunately, the exponentiated gradient’s link function does not satisfy this property. The Mahalanobis distance generating function does not preserve sparsity, neither.

### 3.2.4.4 Discussions

Here is a brief summary of different link functions used in mirror descent, as summarized in Table 3.1, respectively.

Based on the summary above, here we present a brief discussion on the comparison between multiplicative gradient and exponentiated gradient methods. As described above, The two most widely used link functions in mirror descent are the $p$-norm link function [Beck and Teboulle, 2003] and the relative entropy function for exponentiated gradient (EG) [Kivinen and Warmuth, 1995]. Both of these link functions offer a multiplicative update rule compared with regular additive gradient methods. The differences between these two are discussed here. Firstly, the loss function for EG is the relative entropy whereas that of the $p$-norm link function is the squared $l_2$-norm function. Second and more importantly, EG does not produce sparse solutions since it
must maintain the weights away from zero, or else its potential (the relative entropy) becomes unbounded at the boundary.

Another advantage of $p$-norm link functions over EG is that the $p$-norm link function offers a flexible interpolation between additive and multiplicative gradient updates. It has been shown that when the features are dense and the optimal coefficients $\theta^*$ are sparse, EG converges faster than the regular additive gradient methods [Kivinen and Warmuth, 1995]. However, according to our experience, a significant drawback of EG is the overflow of the coefficients due to the exponential operator. To prevent overflow, the most commonly used technique is rescaling: the weights are re-normalized to sum to a constant. However, it seems that this approach does not always work. It has been pointed out [Precup and Sutton, 1997] that in the EG-Sarsa algorithm, rescaling can fail, and replacing eligible traces instead of regular additive eligible traces is used to prevent overflow. EG-Sarsa usually poses restrictions on the basis as well. Thanks to the flexible interpolation capability between multiplicative and additive gradient updates, the $p$-norm link function is more robust and applicable to various basis functions, such as polynomial, radial basis function (RBF), Fourier basis [Konidaris et al., 2011], proto-value functions (PVFs), etc.

### 3.3 Theoretical Analysis

The error bound analysis is given in Appendix.
3.4 Empirical Experiments

3.4.1 Discrete MDPs

Figure 3.1 shows that mirror-descent TD converges more quickly with far smaller Bellman errors than LARS-TD [Kolter and Ng, 2009] based policy iteration (LARS-TD is used for policy evaluation) on a discrete “two-room” MDP [Mahadevan and Maggioni, 2007]. The basis matrix $\Phi$ was automatically generated as 50 proto-value functions by diagonalizing the graph Laplacian of the discrete state space connectivity graph [Mahadevan and Maggioni, 2007]. The figure also shows that Algorithm 5 (sparse mirror-descent TD) scales more gracefully than LARS-TD. Note LARS-TD is unstable for $\gamma = 0.9$. It should be noted that the computation cost of LARS-TD is $O(ndk^2)$, whereas that for Algorithm 5 is $O(nd)$, where $n$ is the number of samples, $d$ is the number of basis functions, and $k$ is the number of active basis functions. If $k$ is linear or sublinear with respect to $d$, Algorithm 5 has a significant advantage over LARS-TD.

Figure 3.2 compares the performance of mirror-descent Q-learning with a fixed $p$-norm link function vs. a decaying $p$-norm link function for a $10 \times 10$ discrete grid world domain with the goal state in the upper left-hand corner. Initially, $p = O(\log d)$ where $d$ is the number of features, and subsequently $p$ is decayed to a minimum of $p = 2$. Varying $p$-norm interpolates between additive and multiplicative updates. Different values of $p$ yield an interpolation between the truncated gradient method [Langford et al., 2009] and SMIDAS [Shalev-Shwartz and Tewari, 2009]. Note that when the $p$-norm link function is decayed, convergence is more rapid.
Figure 3.3 illustrates the performance of Algorithm 6 on the two-room discrete grid world navigation task.

### 3.4.2 Continuous MDPs

Figure 3.4 compares the performance of Q-learning vs. mirror-descent Q-learning for the mountain car task, which converges more quickly to a better solution with much lower variance. Figure 3.5 shows that mirror-descent Q-learning with learned diffusion wavelet bases converges quickly on the 4-dimensional Acrobot task. We found in our experiments that LARS-TD did not converge within 20 episodes (its curve, not shown in Figure 3.4, would be flat on the vertical axis at 1000 steps).

Finally, we tested the mirror-descent approach on a more complex 8-dimensional continuous MDP. The triple-link inverted pendulum [Si and Wang, 2001] is a highly nonlinear time-variant under-actuated system, which is a standard benchmark testbed in the control community. We base our simulation using the system parameters described in [Si and Wang, 2001], except that the action space is discretized because the algorithms described here are restricted to policies with discrete actions. There are three actions, namely \{0,5\text{Newton}, -5\text{Newton}\}. The state space is 8-dimensional, consisting of the angles made to the horizontal of the three links in the arm as well as their angular velocities, the position and velocity of the cart used to balance the pendulum. The goal is to learn a policy that can balance the system with the minimum number of episodes. A run is successful if it balances the inverted pendulum for the specified number of steps within 300 episodes, resulting in a reward of 0. Otherwise, this run is considered as a failure and yields a negative reward $-1$. The first action is chosen randomly to push the pendulum away from the initial state.
Two experiments were conducted on the triple-link pendulum domain with 20 runs for each experiment. As Table 3.2 shows, Mirror Descent Q-learning can be used to learn the policy with fewer episodes and usually with reduced variance compared with regular Q-learning.

The experiment settings are Experiment 1: Zero initial state and the system receives a reward 1 if it is able to balance 10,000 steps. Experiment 2: Zero initial state and the system receives a reward 1 if it is able to balance 100,000 steps. Table 3.2 shows the comparison result between regular Q-learning and Mirror Descent Q-learning.

<table>
<thead>
<tr>
<th># of Episodes \ Experiment</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-learning</td>
<td>6.1 ± 5.67</td>
<td>15.4 ± 11.33</td>
</tr>
<tr>
<td>Mirror Descent Q-learning</td>
<td>5.7 ± 9.70</td>
<td>11.8 ± 6.86</td>
</tr>
</tbody>
</table>

Table 3.2. Results on Triple-Link Inverted Pendulum Task

3.5 Summary

We proposed a novel framework for RL using mirror-descent online convex optimization. Mirror Descent Q-learning demonstrates the following advantage over regular Q-learning: faster convergence rate and reduced variance due to larger stepsizes with theoretical convergence guarantees [Nemirovski et al., 2009]. Compared with existing sparse RL algorithms such as LARS-TD, Algorithm 2 has lower sample complexity and lower computation cost, advantages accrued from the first-order mirror descent framework combined with proximal mapping [Shalev-Shwartz and Tewari, 2011].
Figure 3.1. Mirror-descent Q-learning converges significantly faster than LARS-TD on a “two-room” grid world MDP for $\gamma = 0.9$ (top left) and $\gamma = 0.8$ (top right). The y-axis measures the $l_2$ (red curve) and $l_\infty$ (blue curve) norm difference between successive weights during policy iteration. Bottom: running times for LARS-TD (blue solid) and mirror-descent Q (red dashed). Regularization $\rho = 0.01$. 

51
Figure 3.2. Top figure: convergence of mirror-descent Q-learning with a fixed $p$-norm link function. Bottom figure: decaying $p$-norm link function.
Figure 3.3. Top: Convergence of AdaGrad Mirror-descent Q-learning on two-room gridworld domain. Bottom: Approximated value function, using 50 proto-value function bases
Figure 3.4. Top: Q-learning; Bottom: mirror-descent Q-learning with $p$-norm link function, both with 25 fixed Fourier bases [Konidaris et al., 2011] for the mountain car task.
Figure 3.5. Mirror-descent Q-learning on the Acrobot task using automatically generated diffusion wavelet bases averaged over 5 trials.
CHAPTER 4

REGULARIZED OFF-POLICY TD LEARNING

In the previous chapter we proposed a (on-policy convergent) sparse TD algorithm. Although TD converges when samples are drawn “on-policy” by sampling from the Markov chain underlying a policy in a Markov decision process (MDP), it can be shown to be divergent when samples are drawn “off-policy.”

In this chapter, the off-policy TD learning problem is formulated from the stochastic optimization perspective. A novel objective function is proposed based on the linear inverse problem formulation of the TD learning with correction [Sutton et al., 2009] (TDC) algorithm. The optimization problem underlying off-policy TD methods, such as TDC, is reformulated as a convex-concave saddle-point stochastic approximation problem, which is both convex and incrementally solvable. A detailed theoretical and experimental study of the regularized off-policy temporal difference learning (RO-TD) algorithm is presented [Liu et al., 2012].

4.1 Introduction

4.1.1 Off-Policy Reinforcement Learning

Off-policy learning refers to learning about one way of behaving, called the target policy, from sample sets that are generated by another policy of choosing actions,
which is called the behavior policy, or exploratory policy. As pointed out in [Maei, 2011], the target policy is often a deterministic policy that approximates the optimal policy, and the behavior policy is often stochastic, exploring all possible actions in each state as part of finding the optimal policy. Learning the target policy from the samples generated by the behavior policy allows a greater variety of exploration strategies to be used. It also enables learning from training data generated by unrelated controllers, including manual human control, and from previously collected data. Another reason for interest in off-policy learning is that it enables learning about multiple target policies (e.g., optimal policies for multiple sub-goals) from a single exploratory policy generated by a single behavior policy, which triggered an interesting research area termed as “parallel RL.” Besides, off-policy methods are of wider applications since they can be used to learn while executing an exploratory policy, learn from demonstrations, and learn multiple tasks in parallel [Degris et al., 2012]. Sutton et al. [2009] introduced convergent off-policy TD algorithms, such as TDC, whose computation time scales linearly with the number of samples and the number of features. Recently, a linear off-policy actor-critic algorithm based on the same framework was proposed in Degris et al. [2012].

4.1.2 Convex-concave Saddle-point First-order Algorithms

The key novel contribution of this chapter is a convex-concave saddle-point formulation for regularized off-policy TD learning. A convex-concave saddle-point problem is formulated as follows. Let \( x \in X, y \in Y \), where \( X, Y \) are both nonempty bounded closed convex sets, and \( f(x) : X \to \mathbb{R} \) be a convex function. If there exists a function \( \varphi(\cdot, \cdot) \) such that \( f(x) \) can be represented as \( f(x) := \sup_{y \in Y} \varphi(x, y) \), then the pair
$(\varphi, Y)$ is referred as the saddle-point representation of $f$. The optimization problem of minimizing $f$ over $X$ is converted into an equivalent convex-concave saddle-point problem $\text{SadVal} = \inf_{x \in X} \sup_{y \in Y} \varphi(x, y)$ of $\varphi$ on $X \times Y$. If $f$ is non-smooth yet convex and well structured, which is not suitable for many existing optimization approaches requiring smoothness, its saddle-point representation $\varphi$ is often smooth and convex. Thus, convex-concave saddle-point problems are, therefore, usually better suited for first-order methods [Juditsky and Nemirovski, 2011]. A comprehensive overview on extending convex minimization to convex-concave saddle-point problems with unified variational inequalities is presented in [Ben-Tal and Nemirovski, 2005]. As an example, consider $f(x) = \|Ax - b\|_m$ which admits a bilinear minimax representation

$$f(x) := \|Ax - b\|_m = \max_{\|y\|_n \leq 1} y^\top (Ax - b) \quad (4.1)$$

where $m, n$ are conjugate numbers. Using the approach in [Nemirovski et al., 2009], Equation (4.1) can be solved as

$$x_{t+1} = x_t - \alpha_t A^\top y_t, y_{t+1} = \Pi_n(y_t + \alpha_t (Ax_t - b)) \quad (4.2)$$

where $\Pi_n$ is the projection operator of $y$ onto the unit $l_n$-ball $\|y\|_n \leq 1$, which is defined as

$$\Pi_n(y) = \min(1, 1/\|y\|_n)y, n = 2, 3, \cdots, \Pi_\infty(y_i) = \min(1, 1/|y_i|)y_i \quad (4.3)$$

and $\Pi_\infty$ is an entrywise operator.
4.2 Problem Formulation

Many TD algorithms can be reduced to solving a linear equation, i.e., a linear inverse problem [Ávila Pires and Szepesvári, 2012]. In this section, we first introduce the linear inverse problem formulation of the GTD algorithm family, especially the temporal difference with correction (TDC). Then based on the linear inverse problem formulation, we introduce two types of objective functions, differing in un-squared loss function and squared loss function.

4.2.1 Linear Inverse Problem Formulation

Now let’s review the concept of mean-square projected Bellman error (MSPBE), which is the motivation of the linear inverse problem of TDC algorithm. MSPBE is defined as

\[
\text{MSPBE}(\theta) = \|\Phi \theta - \Pi T(\Phi \theta)\|_\Xi^2
\]

\[
= (\Phi^\top \Xi (T \Phi \theta - \Phi \theta))^\top (\Phi^\top \Xi \Phi)^{-1} \Phi^\top \Xi (T \Phi \theta - \Phi \theta) \tag{4.4}
\]

\[
= \mathbb{E}[\delta_t(\theta) \phi_t]^\top \mathbb{E}[\phi_t \phi_t^\top]^{-1} \mathbb{E}[\delta_t(\theta) \phi_t]
\]

To avoid computing the inverse matrix \((\Phi^\top \Xi \Phi)^{-1}\) and to avoid the double sampling problem [Sutton and Barto, 1998] in (4.4), an auxiliary variable \(w\) is defined

\[
w = \mathbb{E}[\phi_t \phi_t^\top]^{-1} \mathbb{E}[\delta_t(\theta) \phi_t] = (\Phi^\top \Xi \Phi)^{-1} \Phi^\top \Xi (T \Phi \theta - \Phi \theta) \tag{4.5}
\]

Thus we can have the following linear inverse problem

\[
\mathbb{E}[\delta_t(\theta) \phi_t] = \mathbb{E}[\phi_t \phi_t^\top] w = (\Phi^\top \Xi \Phi)w = \Phi^\top \Xi (T \Phi \theta - \Phi \theta) \tag{4.6}
\]
By taking gradient with respect to $\theta$ for optimum condition $\nabla_{\theta} \text{MSPBE}(\theta) = 0$ and utilizing Equation (4.5), we have

$$\mathbb{E}[\delta_t(\theta)\phi_t] = \gamma \mathbb{E}[\phi_t'\phi_t^\top] w$$ (4.7)

Rearranging the two equality of Equation (4.6,4.7), we have the following linear system equation

$$\begin{bmatrix}
\eta \Phi^\top \Xi \Phi & \eta \Phi^\top \Xi (\Phi - \gamma \Phi') \\
\gamma \Phi'^\top \Xi \Phi & \Phi'^\top \Xi (\Phi - \gamma \Phi')
\end{bmatrix}
\begin{bmatrix}
w \\
\theta
\end{bmatrix}
= \begin{bmatrix}
\eta \Phi^\top \Xi R \\
\Phi'^\top \Xi R
\end{bmatrix}$$ (4.8)

The stochastic approximation version of the above equation is as follows, where

$$A = \mathbb{E}[A_t], b = \mathbb{E}[b_t], x = [w; \theta]$$ (4.9)

$$A_t = \begin{bmatrix}
\eta \phi_t' \phi_t^\top & \eta \phi_t (\phi_t' - \gamma \phi_t')^\top \\
\gamma \phi_t' \phi_t^\top & \phi_t (\phi_t' - \gamma \phi_t')^\top
\end{bmatrix}, b_t = \begin{bmatrix}
\eta r_t \phi_t \\
r_t \phi_t
\end{bmatrix}$$ (4.10)

Following Sutton et al. [2009], the TDC algorithm solution follows from the linear equation $Ax = b$, where a single iteration gradient update would be

$$x_{t+1} = x_t - \alpha_t (Ax_t - b_t)$$

where $x_t = [w_t; \theta_t]$. The two time-scale gradient descent learning method TDC [Sutton et al., 2009] is

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi_t - \alpha_t \gamma \phi_t' (\phi_t^\top w_t), w_{t+1} = w_t + \beta_t (\delta_t - \phi_t^\top w_t) \phi_t$$ (4.11)
where $-\alpha_t \gamma \phi_t'(\phi_t^T w_t)$ is the term for correction of gradient descent direction, and $\beta_t = \eta \alpha_t, \eta > 1$.

- $\Xi$ is a diagonal matrix whose entries $\xi(s)$ are given by a positive probability distribution over states. $\Pi = \Phi(\Phi^T \Xi \Phi)^{-1} \Phi^T \Xi$ is the weighted least-squares projection operator.
- A square root of $A$ is a matrix $B$ satisfying $B^2 = A$ and $B$ is denoted as $A^{1/2}$. Note that $A^{1/2}$ may not be unique.
- $[\cdot, \cdot]$ is a row vector, and $[\cdot; \cdot]$ is a column vector.
- For the $t$-th sample, $\phi_t$ (the $t$-th row of $\Phi$), $\phi_t'$ (the $t$-th row of $\Phi'$) are the feature vectors corresponding to $s_t, s'_t$, respectively. $\theta_t$ is the coefficient vector for $t$-th sample in first-order TD learning methods, and $\delta_t = (r_t + \gamma \phi_t'^T \theta_t) - \phi_t^T \theta_t$ is the temporal difference error. Also, $x_t = [w_t; \theta_t], \alpha_t$ is a stepsize, $\beta_t = \eta \alpha_t, \eta > 0$.
- $m, n$ are conjugate numbers if $\frac{1}{m} + \frac{1}{n} = 1, m \geq 1, n \geq 1$. $||x||_m = (\sum_j |x_j|^m)^{\frac{1}{m}}$ is the $m$-norm of vector $x$.
- $\rho$ is $\ell_1$ regularization parameter, $\lambda$ is the eligibility trace factor, $n$ is the sample size, $d$ is the number of basis functions, $k$ is the number of active basis functions.

**Figure 4.1.** Notation and Definitions

### 4.2.2 Un-squared Loss Formulation

There are some issues regarding the objective function, which arise from the online convex optimization and RL perspectives, respectively. The first concern is that the objective function should be convex and stochastically solvable. Note that $A, A_t$ are neither PSD nor symmetric, and it is not straightforward to formulate a convex ob-
jective function based on them. The second concern is that since we do not have knowledge of $A$, the objective function should be separable so that it is stochastically solvable based on $A_t, b_t$. The other concern regards the sampling condition in temporal difference learning: double-sampling. As pointed out in [Sutton and Barto, 1998], double-sampling is a necessary condition to obtain an unbiased estimator if the objective function is the Bellman residual or its derivatives (such as projected Bellman residual), in which the product of Bellman error or projected Bellman error metrics are involved. To overcome this sampling condition constraint, the product of TD errors should be avoided in the computation of gradients. Consequently, based on the linear equation formulation in (4.9) and the requirement on the objective function discussed above, we propose a regularized loss function as

$$\begin{align*}
L(x) &= \|Ax - b\|_m + h(x) \\
(4.12)
\end{align*}$$

Here we also enumerate some intuitive objective functions and give a brief analysis on the reasons why they are not suitable for regularized off-policy first-order TD learning. One intuitive idea is to add a sparsity penalty on MSPBE, i.e., $L(\theta) = \text{MSPBE}(\theta) + \rho\|\theta\|_1$. Because of the $\ell_1$ penalty term, the solution to $\nabla L = 0$ does not have an analytical form and is thus difficult to compute. The second intuition is to use the online least squares formulation of the linear equation $Ax = b$. However, since $A$ is not symmetric and positive semi-definite (PSD), $A^{\frac{1}{2}}$ does not exist and thus $Ax = b$ cannot be reformulated as $\min_{x\in X}\|A^{\frac{1}{2}}x - A^{-\frac{1}{2}}b\|_2^2$. Another possible idea is to attempt to find an objective function whose gradient is exactly $A_t x_t - b_t$ and thus the regularized gradient is $\text{prox}_{\alpha h(x_t)}(A_t x_t - b_t)$. However, since $A_t$ is not
symmetric, this gradient does not explicitly correspond to any kind of optimization problem, not to mention a convex one\(^1\).

### 4.2.3 Squared Loss Formulation

It is also worth noting that there exists another formulation of the loss function different from Equation (4.12) with the following convex-concave formulation as in [Nesterov, 2007; Juditsky and Nemirovski, 2011],

\[
\begin{align*}
\min_x \frac{1}{2} \|Ax - b\|_2^2 + \rho \|x\|_1 &= \max_{\|A^T y\|_\infty \leq 1} (b^T y - \frac{\rho}{2} y^T y) \\
&= \min_x \max_{\|u\|_\infty \leq 1, y} \left( x^T u + y^T (Ax - b) - \frac{\rho}{2} y^T y \right) \quad (4.13)
\end{align*}
\]

Here we give the detailed deduction of formulation in Equation (4.13). First, using the dual norm representation, the standard LASSO problem formulation is reformulated as

\[
f(x) = \frac{1}{2} \|Ax - b\|_2^2 + \rho \|x\|_1 = \max_{y, \|A^T y\|_\infty \leq 1} \left[ \langle b/\rho, y \rangle - \frac{1}{2} y^T y \right] \quad (4.14)
\]

Then\(^2\)

\[^1\text{Note that the A matrix in GTD2’s linear equation representation is symmetric, yet is not PSD, so it cannot be formulated as a convex problem.}\]

\[^2\text{Let } w = -y, \text{ then we have the same formulation as in Nemirovski’s tutorial in COLT2012.}\]

\[
\Phi(x, w) = \langle w, Ax - b \rangle - \frac{1}{2} w^T w - \langle x, A^T w \rangle
\]
\[ \langle b, y \rangle - \frac{1}{2} y^\top y = \langle b, y \rangle - \frac{1}{2} y^\top y + \langle x, A^\top y \rangle - \langle y, Ax \rangle \]
\[ = \langle y, b - Ax \rangle - \frac{1}{2} y^\top y + \langle x, A^\top y \rangle \]

which can be solved iteratively without the proximal gradient step as follows, which serves as a counterpart of Equation (4.17),

\[ x_{t+1} = x_t - \alpha_t \rho (u_t + A_t^\top y_t), \quad y_{t+1} = y_t + \frac{\alpha_t}{\rho} (A_t x_t - b_t - \rho y_t) \]
\[ u_{t+\frac{1}{2}} = u_t + \frac{\alpha_t}{\rho} x_t, \quad u_{t+1} = \Pi_\infty (u_{t+\frac{1}{2}}) \quad (4.15) \]

4.3 Algorithm Design

4.3.1 RO-TD Algorithm Design

In this section, the problem of (4.12) is formulated as a convex-concave saddle-point problem, and the RO-TD algorithm is proposed. Analogous to (4.1), the regularized loss function can be formulated as

\[ \|Ax - b\|_m + h(x) = \max_{\|y\|_1 \leq 1} y^\top (Ax - b) + h(x) \quad (4.16) \]

Similar to (4.2), Equation (4.16) can be solved via an iteration procedure as follows, where \( x_t = [w_t; \theta_t] \).

\[ x_{t+\frac{1}{2}} = x_t - \alpha_t A_t^\top y_t, \quad y_{t+\frac{1}{2}} = y_t + \alpha_t (A_t x_t - b_t) \]
\[ x_{t+1} = \text{prox}_{\alpha_t h}(x_{t+\frac{1}{2}}), \quad y_{t+1} = \Pi_n (y_{t+\frac{1}{2}}) \quad (4.17) \]
The averaging step, which plays a crucial role in stochastic optimization convergence, generates the *approximate saddle-points* [Juditsky and Nemirovski, 2011; Nedic and Ozdaglar, 2009]

\[
\bar{x}_t = \left(\sum_{i=0}^{T} \alpha_i \right)^{-1} \sum_{i=0}^{T} \alpha_i x_i, \quad \bar{y}_t = \left(\sum_{i=0}^{T} \alpha_i \right)^{-1} \sum_{i=0}^{T} \alpha_i y_i \tag{4.18}
\]

Due to the computation of \(A_t\) in (4.17) at each iteration, the computation cost appears to be \(O(nd^2)\), where \(n,d\) are defined in Figure 4.1. However, the computation cost is actually \(O(nd)\) with a linear algebraic trick by computing not \(y_t^\top A_t\) but \(y_t^\top A_t x_t - b_t\).

Denoting \(y_t = [y_{1,t}; y_{2,t}]\), where \(y_{1,t}; y_{2,t}\) are column vectors of equal length, we have

\[
y_t^\top A_t = \begin{bmatrix}
\eta \phi_t^\top (y_{1,t}^\top \phi_t) + \gamma \phi_t^\top (y_{2,t}^\top \phi_t') \quad (\phi_t - \gamma \phi_t')^\top (\eta y_{1,t}^\top + y_{2,t}^\top) \phi_t
\end{bmatrix} \tag{4.19}
\]

\(A_t x_t - b_t\) can be computed according to Equation (4.11) as follows:

\[
A_t x_t - b_t = \begin{bmatrix}
-\eta (\delta_t - \phi_t^\top w_t) \phi_t; \quad \gamma (\phi_t^\top w_t) \phi_t' - \delta_t \phi_t
\end{bmatrix} \tag{4.20}
\]

Both (4.19) and (4.20) are of linear computational complexity. Now we are ready to present the RO-TD algorithm:

There are some design details of the algorithm to be elaborated. First, the regularization term \(h(x)\) can be any kind of convex regularization, such as ridge regression or sparsity penalty \(\rho ||x||_1\). In case of \(h(x) = \rho ||x||_1\), \(\text{pro}x_{\alpha h}(\cdot) = S_{\alpha \rho}(\cdot)\). In real applications the sparsification requirement on \(\theta\) and auxiliary variable \(w\) may be different, i.e., \(h(x) = \rho_1 ||\theta||_1 + \rho_2 ||w||_1, \rho_1 \neq \rho_2\), one can simply replace the uniform soft
thresholding $S_{\alpha \rho}$ by two separate soft thresholding operations $S_{\alpha \rho_1}, S_{\alpha \rho_2}$ and thus the third equation in (4.17) is replaced by the following,

$$x_{t+\frac{1}{2}} = \left[ w_{t+\frac{1}{2}}, \theta_{t+\frac{1}{2}} \right], \theta_{t+1} = S_{\alpha \rho_1}(\theta_{t+\frac{1}{2}}), w_{t+1} = S_{\alpha \rho_2}(w_{t+\frac{1}{2}})$$  \hspace{1cm} (4.21)

Another concern is the choice of conjugate numbers $(m, n)$. For ease of computing $\Pi_n$, we use $(2, 2)(l_2 \text{ fit}), (+\infty, 1)(\text{uniform fit})$ or $(1, +\infty)$. $m = n = 2$ is used in the experiments below.

**Algorithm 7 RO-TD**

Let $\pi$ be some fixed policy of an MDP $M$, and let the sample set $S = \{s_i, r_i, s_i'\}_{i=1}^N$. Let $\Phi$ be some fixed basis.

1. **REPEAT**
   2. Compute $\phi_t, \phi_t'$ and TD error $\delta_t = (r_t + \gamma \phi_t'^T \theta_t) - \phi_t^T \theta_t$
   3. Compute $y_t^T A_t, A_t x_t - b_t$ in Equation (4.19) and (4.20).
   4. Compute $x_{t+1}, y_{t+1}$ as in Equation (4.17)
   5. Set $t \leftarrow t + 1$
   6. **UNTIL** $t = N$
   7. Compute $\bar{x}_N, \bar{y}_N$ as in Equation (4.18) with $t = N$

**4.3.2 RO-GQ($\lambda$) Design**

GQ($\lambda$)[Maei and Sutton, 2010] is a generalization of the TDC algorithm with eligibility traces and off-policy learning of temporally abstract predictions, where the gradient update changes from Equation (4.11) to

$$\theta_{t+1} = \theta_t + \alpha_t[\delta_t e_t - \gamma(1 - \lambda)w_t^T e_t \bar{\phi}_{t+1}], w_{t+1} = w_t + \beta_t(\delta_t e_t - w_t^T \phi_t \phi_t)$$  \hspace{1cm} (4.22)
The central element is to extend the MSPBE function to the case where it incorporates eligibility traces. The objective function and corresponding linear equation component $A_t, b_t$ can be written as follows:

$$L(\theta) = \| \Phi \theta - \Pi T^{\pi \lambda} \Phi \theta \|_2^2$$  \hspace{1cm} (4.23)

$$A_t = \begin{bmatrix}
\eta \phi_t \phi_t^T & \eta e_t (\phi_t - \gamma \phi_{t+1})^T \\
\gamma (1 - \lambda) \bar{\phi}_{t+1} e_t^T & e_t (\phi_t - \gamma \bar{\phi}_{t+1})^T 
\end{bmatrix},
 b_t = \begin{bmatrix}
\eta r_t e_t \\
r_t e_t
\end{bmatrix}$$  \hspace{1cm} (4.24)

Similar to Equation (4.19) and (4.20), the computation of $y_t^T A_t, A_t x_t - b_t$ is

$$y_t^T A_t = \begin{bmatrix}
\eta \phi_t (y_1^T \phi_t) + \gamma (1 - \lambda) e_t^T (y_2^T \bar{\phi}_{t+1}) (\phi_t - \gamma \bar{\phi}_{t+1}) (\eta y_1^T + y_2^T) e_t \\
-\eta (\delta e_t - \phi_t^T w_t \phi_t); \gamma (1 - \lambda) (e_t^T w_t) \bar{\phi}_{t+1} - \delta e_t
\end{bmatrix}$$  \hspace{1cm} (4.25)

where eligibility traces $e_t$, and $\bar{\phi}_t, T^{\pi \lambda}$ are defined in [Maei and Sutton, 2010]. Algorithm 8, RO-GQ($\lambda$), extends the RO-TD algorithm to include eligibility traces.

### 4.4 Theoretical Analysis

The theoretical analysis of RO-TD algorithm can be seen in the Appendix.

### 4.5 Empirical Results

We now demonstrate the effectiveness of the RO-TD algorithm against other algorithms across a number of benchmark domains. LARS-TD [Kolter and Ng, 2009],
Algorithm 8 RO-GQ(\(\lambda\))
Let \(\pi\) and \(\Phi\) be as defined in Algorithm 4. Starting from \(s_0\).

1. **REPEAT**
   2. Compute \(\phi_t, \bar{\phi}_{t+1}\) and TD error \(\delta_t = (r_t + \gamma \bar{\phi}_{t+1}^\top \theta_t) - \phi_t^\top \theta_t\)
   3. Compute \(y_t^\top A_t, A_t x_t - b_t\) in Equation (4.25).
   4. Compute \(x_{t+1}, y_{t+1}\) as in Equation (4.17)
   5. Choose action \(a_t\), and get \(s_{t+1}\)
   6. Set \(t \leftarrow t + 1\);
   7. **UNTIL** \(s_t\) is an absorbing state;
   8. Compute \(\bar{x}_t, \bar{y}_t\) as in Equation (4.18)

which is a popular second-order sparse RL algorithm, is used as the baseline algorithm for feature selection and TDC is used as the off-policy convergent RL baseline algorithm, respectively.

4.5.1 MSPBE Minimization and Off-Policy Convergence
This experiment aims to show the minimization of MSPBE and off-policy convergence of the RO-TD algorithm. The 7 state star MDP is a well-known counterexample where TD diverges monotonically and TDC converges. It consists of 7 states and the reward with respect to any transition is zero. Because of this, the star MDP is unsuitable for LSTD-based algorithms, including LARS-TD since \(\Phi^\top R = 0\) always holds. The random-walk problem is a standard Markov chain with 5 states and two absorbing states at two ends. Three sets of different bases \(\Phi\) are used in [Sutton et al., 2009], which are tabular features, inverted features and dependent features respectively. An
Figure 4.2. Illustrative examples of the convergence of RO-TD using the Star and Random-walk MDPs.
identical experiment setting to Sutton et al. [2009] is used for these two domains. The regularization term \( h(x) \) is set to 0 to make a fair comparison with TD and TDC. \( \alpha = 0.01, \eta = 10 \) for TD, TDC and RO-TD. The comparison with TD, TDC and RO-TD is shown in the left sub-figure of Figure 4.2, where TDC and RO-TD have almost identical MSPBE over iterations. The middle sub-figure shows the value of \( y_t^T (Ax_t - b) \) and \( \|Ax_t - b\|_2 \), in which \( \|Ax_t - b\|_2 \) is always greater than the value of \( y_t^T (Ax_t - b) \). Note that for this problem, the Slater condition is satisfied so there is no duality gap between the two curves. As the result shows, TDC and RO-TD perform equally well, which illustrates the off-policy convergence of the RO-TD algorithm. The result of random-walk chain is averaged over 50 runs. The rightmost sub-figure of Figure 4.2 shows that RO-TD is able to reduce MSPBE over successive iterations with respect to three different basis functions.

4.5.2 Feature Selection

In this section, we use the mountain car example with a variety of bases to show the feature selection capability of RO-TD. The Mountain car is an optimal control problem with a continuous two-dimensional state space. The steep discontinuity in the value function makes learning difficult for bases with global support. To make a fair comparison, we use the same basis function setting as in [Kolter and Ng, 2009], where two-dimensional grids of 2, 4, 8, 16, 32 RBFs are used so that there are totally 1365 basis functions. For LARS-TD, 500 samples are used. For RO-TD and TDC, 3000 samples are used by executing 15 episodes with 200 steps for each episode, stepsize \( \alpha_t = 0.001 \), and \( \rho_1 = 0.01, \rho_2 = 0.2 \). We use the result of LARS-TD and \( l_2 \) LSTD reported in [Kolter and Ng, 2009]. As the result shows in Table 4.1, RO-TD is
able to perform feature selection successfully, whereas TDC and TD failed. It is worth noting that comparing the performance of RO-TD and LARS-TD is not the major focus here, since LARS-TD is not convergent off-policy and RO-TD’s performance can be further optimized using the mirror-descent approach with the Mirror-Prox algorithm [Juditsky and Nemirovski, 2011] which incorporates mirror descent with an extragradient [Korpelevich, 1976], as discussed below.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>LARS-TD</th>
<th>RO-TD</th>
<th>$l_2$ LSTD</th>
<th>TDC</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success (20/20)</td>
<td>100%</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Steps</td>
<td>142.25 ± 9.74</td>
<td>147.40 ± 13.31</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 4.1.** Comparison of TD, LARS-TD, RO-TD, $l_2$ LSTD, TDC and TD

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Method</th>
<th>RO-GQ($\lambda$)</th>
<th>GQ($\lambda$)</th>
<th>LARS-TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>6.9 ± 4.82</td>
<td>11.3 ± 9.58</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Experiment 2</td>
<td>14.7 ± 10.70</td>
<td>27.2 ± 6.52</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.2.** Comparison of RO-GQ($\lambda$), GQ($\lambda$), and LARS-TD on Triple-Link Inverted Pendulum Task

### 4.5.3 High-dimensional Under-actuated Systems

The triple-link inverted pendulum [Si and Wang, 2001] is a highly nonlinear under-actuated system with 8-dimensional state space and discrete action space. The state space consists of the angles and angular velocity of each arm as well as the position and velocity of the car. The discrete action space is \{0, 5Newton, −5Newton\}. The goal is to learn a policy that can balance the arms for $N_x$ steps within some minimum number of learning episodes. The allowed maximum number of episodes is 300. The pendulum initiates from zero equilibrium state and the first action is randomly chosen.
to push the pendulum away from the initial state. We test the performance of RO-GQ(\(\lambda\)), GQ(\(\lambda\)) and LARS-TD. Two experiments are conducted with \(N_x = 10,000\) and 100,000, respectively. Fourier basis [Konidaris et al., 2011] with order 2 is used, resulting in 6561 basis functions. Table 4.2 shows the results of this experiment, where RO-GQ(\(\lambda\)) performs better than other approaches, especially in Experiment 2, which is a harder task. LARS-TD failed in this domain, which is mainly not due to LARS-TD itself but the quality of samples collected via random walk.

To sum up, RO-GQ(\(\lambda\)) tends to outperform GQ(\(\lambda\)) in all aspects, and is able to outperform LARS-TD based policy iteration in high-dimensional domains, as well as in selected smaller MDPs where LARS-TD diverges (e.g., the star MDP). It is worth noting that the computation cost of LARS-TD is \(O(ndk^2)\), where that for RO-TD is \(O(nd)\). If \(k\) is linear or sublinear with respect to \(d\), RO-TD has a significant advantage over LARS-TD. However, compared with LARS-TD, RO-TD requires fine tuning the parameters of \(\alpha_t, \rho_1, \rho_2\) and is usually not as sample efficient as LARS-TD. We also find that tuning the sparsity parameter \(\rho_2\) generates an interpolation between GQ(\(\lambda\)) and Q-learning, where a large \(\rho_2\) helps eliminate the correction term of TDC update and make the update direction more similar to the TD update.

4.6 Summary

In this chapter, we presented a novel unified framework for designing regularized off-policy convergent RL algorithms combining a convex-concave saddle-point problem formulation for RL with stochastic first-order methods. A detailed experimental
analysis reveals that the proposed RO-TD algorithm is both off-policy convergent and robust to noisy features.
CHAPTER 5

FINITE-SAMPLE ANALYSIS OF PROXIMAL GRADIENT TD ALGORITHMS

In this chapter, we show how gradient TD (GTD) methods can be formally derived as true stochastic gradient algorithms [Liu et al., 2015], not with respect to their original objective functions as previously attempted, but rather using derived primal-dual saddle-point objective functions. We then conduct a saddle-point error analysis to obtain finite-sample bounds on their performance. Previous analyses of this class of algorithms use stochastic approximation techniques to prove asymptotic convergence, and no finite-sample analysis had been successfully conducted. Two novel GTD algorithms are also proposed, namely projected GTD2 and GTD2-MP, which use proximal “mirror maps” to yield improved convergence guarantees and acceleration, respectively. The results of our theoretical analysis imply that the GTD family of algorithms are comparable and may indeed be preferred over existing least-squares TD methods for off-policy learning, due to their linear complexity. We provide experimental results showing the improved performance of our accelerated gradient TD methods.
5.1 Introduction

Obtaining a true stochastic gradient temporal difference method has been a long-standing goal of RL [Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998] ever since it was discovered that the original TD method was unstable in many off-policy scenarios where the target behavior being learned and the exploratory behavior producing samples differ. Sutton et al. [2008, 2009] proposed the family of gradient-based temporal difference (GTD) algorithms which offer several interesting properties. A key property of this class of GTD algorithms is that they are asymptotically off-policy convergent, which was shown using stochastic approximation [Borkar, 2008]. This is quite important when we notice that many RL algorithms, especially those that are based on stochastic approximation, such as TD(λ), do not have convergence guarantees in the off-policy setting.

Unfortunately, the aforementioned GTD algorithms are not true stochastic gradient methods with respect to their original objective functions, as pointed out in [Szepesvári, 2010]. The reason is not surprising: the gradient of the objective functions used involve products of terms, which cannot be sampled directly, and was decomposed by a rather ad-hoc splitting of terms. In this chapter, we take a major step forward in resolving this problem by showing a principled way of designing true stochastic gradient TD algorithms by using a primal-dual saddle point objective function, derived from the original objective functions, coupled with the principled use of operator splitting [Bauschke and Combettes, 2011]. An appealing property of these algorithms is their first-order computational complexity that allows them to scale more gracefully to high-dimensional problems, unlike the widely used least-squares TD (LSTD) approaches [Bradtke and Barto, 1996] that only perform well with moderate-sized RL
problems, due to their quadratic (with respect to the dimension of the feature space) computational cost per iteration.

Finite-sample analysis of an algorithm is used to give a full characterization of the performance of the algorithm when only a finite number of samples is available (which is the most common situation in practice). Compared with asymptotic analysis, a finite-sample analysis of a policy evaluation algorithm has a number of important advantages: 1) unlike the assumptions made in the asymptotic analysis of a policy evaluation algorithm, where they assume that algorithm always returns a solution, in a finite-sample analysis we study the characteristics of the actual empirical performance of the algorithm’s solution, including its existence, 2) a finite-sample bound explicitly reveals how the prediction error of the algorithm is related to the characteristic parameters of the MDP at hand, such as the discount factor, the dimensionality of the function space, and the number of samples, 3) once this dependency is clear, the bound can be used to determine the order of magnitude of the number of samples needed to achieve a desired accuracy.

Since in real-world applications of RL, we have access to only a finite amount of data, finite-sample analysis of gradient TD algorithms is essential as it clearly shows the effect of the number of samples (and the parameters that play a role in the sampling budget of the algorithm) on their final performance. However, most of the work on finite-sample analysis in RL has been focused on batch RL (or approximate dynamic programming) algorithms (e.g., Kakade and Langford 2002; Munos and Szepesvári 2008; Antos et al. 2008; Lazaric et al. 2010a), especially those that are least squares TD (LSTD)-based (e.g., Lazaric et al. 2010b; Ghavamzadeh et al. 2010, 2011; Lazaric et al. 2012), and more importantly restricted to the on-policy setting. In this chapter,
we provide the finite-sample analysis of the GTD family of algorithms, a relatively novel class of gradient-based TD methods that are guaranteed to converge even in the off-policy setting, and for which, to the best of our knowledge, no finite-sample analysis has been reported. This analysis is challenging because 1) the stochastic approximation methods that have been used to prove the asymptotic convergence of these algorithms do not address convergence rate analysis; 2) as we explain in detail in Section 5.2.1, the techniques used for the analysis of the stochastic gradient methods cannot be applied here; 3) finally, the difficulty of finite-sample analysis in the off-policy setting. It should also be noted that there exists very little literature on finite-sample analysis in the off-policy setting, even for the LSTD-based algorithms that have been extensively studied.

The major contributions of this chapter include

- the first finite-sample analyses of TD algorithms with linear computational complexity, which is also one of the first few finite-sample analyses of off-policy convergent TD algorithms.

- a novel framework for designing gradient-based TD algorithms with Bellman Error based objective functions, as well as the design and analysis of several improved GTD methods that result from our novel approach of formulating gradient TD methods as true stochastic gradient algorithms with respect to a saddle-point objective function.

We then use the techniques applied in the analysis of the stochastic gradient methods to propose a unified finite-sample analysis for the previously proposed as well as our novel gradient TD algorithms. Finally, given the results of our analysis, we study the
GTD class of algorithms from several different perspectives, including acceleration in convergence, learning with biased importance sampling factors, etc. It should be noted that the developed algorithms are all for the policy evaluation of a fixed policy. The control learning extension of these algorithm can be conducted in a similar way as the Greedy-GQ algorithm [Maei and Sutton, 2010].

5.2 Preliminaries

In this section, we introduce some preliminaries of gradient-based TD learning, and related work along this direction.

5.2.1 Gradient-based TD Algorithms

The class of gradient-based TD (GTD) algorithms was proposed by Sutton et al. [2008, 2009]. These algorithms target two objective functions: the norm of the expected TD update (NEU) and the mean-square projected Bellman error (MSPBE), defined as (see e.g., Maei 2011)\(^1\)

\[
\text{NEU}(\theta) = || \Phi^\top \Xi (T\hat{v} - \hat{v}) ||^2, \\
\text{MSPBE}(\theta) = || \hat{v} - \Pi T \hat{v} ||^2_2 = || \Phi^\top \Xi (T\hat{v} - \hat{v}) ||^2_{C^{-1}},
\]

where \(C = \mathbb{E}[\phi_i \phi_i^\top] = \Phi^\top \Xi \Phi\) is the covariance matrix defined in Eq. 2.3 and is assumed to be non-singular, and \(\Pi = \Phi (\Phi^\top \Xi \Phi)^{-1} \Phi^\top \Xi\) is the orthogonal projection operator onto the function space \(\mathcal{F}\), i.e., for any bounded function \(g\), \(\Pi g = \arg \min_{f \in \mathcal{F}} || g - f ||_\Xi\).

\(^1\)It is important to note that \(T\) in (5.1) and (5.2) is \(T^\pi\), the Bellman operator of the target policy \(\pi\).
From (5.1) and (5.2), it is clear that NEU and MSPBE are squared unweighted and weighted by $C^{-1}$, $\ell_2$-norms of the quantity $\Phi^\top \Xi (T\hat{v} - \hat{v})$, respectively, and thus, the two objective functions can be unified as

$$J(\theta) = \|(\Phi^\top \Xi (T\hat{v} - \hat{v}))\|_{M^{-1}}^2 = \||E[\rho_i \delta_i(\theta)\phi_i]|\|_{M^{-1}}^2,$$

with $M$ equals to the identity matrix $I$ for NEU and to the covariance matrix $C$ for MSPBE. The second equality in (5.3) holds because of the following lemma from Section 4.2 in [Maei, 2011].

We denote by $\pi_b$, the behavior policy that generates the data, and by $\pi$, the target policy that we would like to evaluate. They are the same in the on-policy setting and different in the off-policy setting. For each state-action pair $(s_i, a_i)$, such that $\pi_b(a_i|s_i) > 0$, we define the importance-weighting factor $\rho_i = \pi(a_i|s_i)/\pi_b(a_i|s_i)$ with $\rho_{\text{max}} \geq 0$ being its maximum value over the state-action pairs.

Finally, we define the matrices $A$, and the vector $b$ as

$$A := E[\rho_i \phi_i (\Delta \phi_i)^\top], \quad b := E[\rho_i \phi_i r_i]$$

(5.4)

where the expectations are with respect to $\xi$ and $P^{ab}$. We also denote by $\Xi$, the diagonal matrix whose elements are $\xi(s)$, and $\xi_{\text{max}} := \max_s \xi(s)$. For each sample $i$ in the training set $D$, we can calculate unbiased estimates of $A$, $b$, and $C$ as follows:

$$\hat{A}_i := \rho_i \phi_i \Delta \phi_i^\top, \quad \hat{b}_i := \rho_i r_i \phi_i, \quad \hat{C}_i := \phi_i \phi_i^\top.$$ 

(5.5)
Note that compared with the previous definitions of $A, b, \hat{A}_i, \hat{b}_i$, the definitions of $A, b, \hat{A}_i, \hat{b}_i$ include the importance-weighting factor $\rho_i$.

**Lemma 6.** (Importance-weighting for off-policy TD) Let $D = \{(s_i, a_i, r_i, s_i')\}_{i=1}^n$, $s_i \sim \xi, a_i \sim \pi_b(\cdot|s_i), s_i' \sim P(\cdot|s_i, a_i)$ be a training set generated by the behavior policy $\pi_b$ and $T$ be the Bellman operator of the target policy $\pi$. Then, we have

$$\Phi^\top \Xi (T\hat{v} - \hat{v}) = \mathbb{E}[\rho_i \delta_i(\theta) \phi_i] = b - A\theta.$$

**Proof.** We give a proof sketch here. Refer to Section 4.2 in [Maei, 2011] for a detailed proof.

$$\Phi^\top \Xi (T\hat{v} - \hat{v})$$

$$= \sum_{s,a,s'} \xi(s) \pi(a|s) P(s'|s,a) \delta(\theta|s,a,s') \phi(s)$$

$$= \sum_{s,a,s'} \xi(s) \frac{\pi(a|s)}{\pi_b(a|s)} P_b(s'|s,a) \delta(\theta|s,a,s') \phi(s)$$

$$= \sum_{s,a,s'} \xi(s) \rho(s,a) \pi_b(s'|s,a) \delta(\theta|s,a,s') \phi(s)$$

$$= \mathbb{E}[\rho_i \delta_i(\theta) \phi_i]$$

$$= b - A\theta$$

\[\square\]

Motivated by minimizing the NEU and MSPBE objective functions using the stochastic gradient methods, the GTD and GTD2 algorithms were proposed with the following update rules:
\[ GTD:\quad y_{t+1} = y_t + \alpha_t (\rho_t \delta_t (\theta_t) \phi_t - y_t), \quad (5.6) \]
\[ \theta_{t+1} = \theta_t + \alpha_t \rho_t \Delta \phi_t (y_t^\top \phi_t), \]
\[ GTD2:\quad y_{t+1} = y_t + \alpha_t (\rho_t \delta_t (\theta_t) - \phi_t^\top y_t) \phi_t, \quad (5.7) \]
\[ \theta_{t+1} = \theta_t + \alpha_t \rho_t \Delta \phi_t (y_t^\top \phi_t). \]

However, it has been shown that the above update rules do not update the value function parameter \( \theta \) in the gradient direction of NEU and MSPBE, and thus, NEU and MSPBE are not the true objective functions of the GTD and GTD2 algorithms [Szepesvári, 2010]. Consider the NEU objective function in (5.1). Taking its gradient with respect to \( \theta \), we obtain
\[
\frac{1}{2} \nabla \text{NEU}(\theta) = -\left( \nabla \mathbb{E}[\rho_i \delta_i (\theta) \phi_i^\top] \right) \mathbb{E}[\rho_i \delta_i (\theta) \phi_i]
= -\left( \mathbb{E}[\rho_i \nabla \delta_i (\theta) \phi_i^\top] \right) \mathbb{E}[\rho_i \delta_i (\theta) \phi_i]
= \mathbb{E}[\rho_i \Delta \phi_i \phi_i^\top] \mathbb{E}[\rho_i \delta_i (\theta) \phi_i]. \quad (5.8)
\]

If the gradient can be written as a single expectation, then it is straightforward to use a stochastic gradient method. However, we have a product of two expectations in (5.8), and unfortunately, due to the correlation between them, the sample product (with a single sample) won’t be an unbiased estimate of the gradient. To tackle this, the GTD algorithm uses an auxiliary variable \( y_t \) to estimate \( \mathbb{E}[\rho_i \delta_i (\theta) \phi_i] \), and thus, the overall algorithm is no longer a true stochastic gradient method with respect to NEU. It can be easily shown that the same problem exists for GTD2 with respect to
the MSPBE objective function. This prevents us from using the standard convergence analysis techniques of stochastic gradient descent methods to obtain a finite-sample performance bound for the GTD and GTD2 algorithms.

It should be also noted that in the original publications of GTD/GTD2 algorithms [Sutton et al., 2008, 2009], the authors discussed handling the off-policy scenario using both importance and rejection sampling. In rejection sampling, which was mainly used in [Sutton et al., 2008, 2009], a sample \((s_i, a_i, r_i, s'_i)\) is rejected and the parameter \(\theta\) is not updated if \(\pi(a_i|s_i) = 0\). This sampling strategy is not efficient since a lot of samples will be discarded if \(\pi_b\) and \(\pi\) are very different.

### 5.2.2 Related Work

Before we present a finite-sample performance bound for GTD and GTD2, it is helpful to give a brief overview of the existing literature on finite-sample analysis of the TD algorithms. The convergence rate of the TD algorithms mainly depends on \((d, n, \nu)\), where \(d\) is the size of the approximation space (the dimension of the feature vector), \(n\) is the number of samples, and \(\nu\) is the smallest eigenvalue of the sample-based covariance matrix \(\hat{C} = \hat{\Phi}^\top \hat{\Phi}\), i.e., \(\nu = \lambda_{\min}(\hat{C})\).

Antos et al. [2008] proved an error bound of \(O\left(\frac{d \log d}{n^{1/4}}\right)\) for LSTD in bounded spaces. Lazaric et al. [2010b] proposed a LSTD analysis in learner spaces and obtained a tighter bound of \(O\left(\sqrt{\frac{d \log d}{n \nu}}\right)\) and later used it to derive a bound for the least-squares policy iteration (LSPI) algorithm [Lazaric et al., 2012]. Tagorti and Scherrer [2014] recently proposed the first convergence analysis for LSTD(\(\lambda\)) and derived a bound of \(\tilde{O}(d/\nu \sqrt{n})\). The analysis is a bit different than the one in [Lazaric et al., 2010b] and the bound is weaker in terms of \(d\) and \(\nu\). Another recent result is by Prashanth et al. [2014] that
uses stochastic approximation to solve LSTD(0), where the resulting algorithm is exactly TD(0) with random sampling (samples are drawn i.i.d. and not from a trajectory), and report a Markov design bound (the bound is computed only at the states used by the algorithm) of $O(\sqrt{d/n})$ for LSTD(0). All these results are for the on-policy setting, except the one by Antos et al. [2008] that also holds for the off-policy formulation. Another result in the off-policy setting is by Ávila Pires and Szepesvári [2012] that uses a bounding trick and improves the result of Antos et al. [2008] by a log $d$ factor. Another line of work is by Yu [2012], which provides error bounds of LSTD algorithms for a wide range of problems including the scenario that $||A||_\infty$ is unbounded, which is beyond the scope of the aforementioned literature and this thesis.

The line of research reported here has much in common with work on proximal RL [Mahadevan et al., 2014], which explores first-order RL algorithms using mirror maps [Bubeck, 2014; Juditsky et al., 2008] to construct primal-dual spaces. This work began originally with a dual space formulation of first-order sparse TD learning [Mahadevan and Liu, 2012]. A saddle point formulation for off-policy TD learning was initially explored in [Liu et al., 2012], where the objective function is the norm of the approximation residual of a linear inverse problem [Ávila Pires and Szepesvári, 2012]. A sparse off-policy GTD2 algorithm with regularized dual averaging is introduced by Qin and Li [2014]. These studies provide different approaches to formulating the problem 1) as a variational inequality problem [Juditsky et al., 2008; Mahadevan et al., 2014], 2) as a linear inverse problem [Liu et al., 2012], or 3) as a quadratic objective function (MSPBE) using two-time-scale solvers [Qin and Li, 2014]. In this chapter, we explore the true nature of the GTD algorithms as stochastic gradient
algorithms with respect to the convex-concave saddle-point formulations of NEU and MSPBE.

5.3 Saddle-point Formulation Of GTD Algorithms

In this section, we show how the GTD and GTD2 algorithms can be formulated as true stochastic gradient (SG) algorithms by writing their respective objective functions, NEU and MSPBE, in the form of a convex-concave saddle-point. As discussed earlier, this new formulation of GTD and GTD2 as true SG methods allows us to use the convergence analysis techniques for SGs in order to derive finite-sample performance bounds for these RL algorithms. Moreover, it allows us to use more efficient algorithms that have been recently developed to solve SG problems, such as stochastic Mirror-Prox (SMP) [Juditsky et al., 2008], to derive more efficient versions of GTD and GTD2.

A particular type of convex-concave saddle-point formulation is formally defined as

\[
\min_{\theta} \max_{y} (L(\theta, y) = \langle b - A\theta, y \rangle + F(\theta) - K(y)),
\]

(5.9)

where \( F(\theta) \) is a convex function and \( K(y) \) is a smooth convex function such that

\[
K(y) - K(x) - \langle \nabla K(x), y - x \rangle \leq \frac{L_K}{2} ||x - y||^2.
\]

(5.10)

Next we follow Juditsky et al. [2008]; Nemirovski et al. [2009]; Chen et al. [2013] and define the following error function for the saddle-point problem (5.9).
Definition 12. The error function of the saddle-point problem (5.9) at each point \((\theta', y')\) is defined as

\[
\text{Err}(\theta', y') = \max_y L(\theta', y) - \min_\theta L(\theta, y').
\] (5.11)

In this chapter, we consider the saddle-point problem (5.9) with \(F(\theta) = 0\) and \(K(y) = \frac{1}{2}||y||_M^2\), i.e.,

\[
\min_\theta \max_y \left( L(\theta, y) = \langle b - A\theta, y \rangle - \frac{1}{2}||y||_M^2 \right),
\] (5.12)

where \(A\) and \(b\) were defined by Eq. 5.4, and \(M\) is a positive definite matrix. It is easy to show that \(K(y) = \frac{1}{2}||y||_M^2\) satisfies the condition in Eq. 5.10.

We first show in Proposition 1 that if \((\theta^*, y^*)\) is the saddle-point of problem (5.12), then \(\theta^*\) will be the optimum of NEU and MSPBE defined in Eq. 5.3. We then prove in Proposition 2 that GTD and GTD2 in fact find this saddle-point.

Proposition 1. For any fixed \(\theta\), we have \(\frac{1}{2}J(\theta) = \max_y L(\theta, y)\), where \(J(\theta)\) is defined by Eq. 5.3.

Proof. Since \(L(\theta, y)\) is an unconstrained quadratic program with respect to \(y\), the optimal \(y^*(\theta) = \arg \max_y L(\theta, y)\) can be analytically computed as

\[
y^*(\theta) = M^{-1}(b - A\theta).
\] (5.13)

The result follows by plugging \(y^*\) into (5.12) and using the definition of \(J(\theta)\) in Eq. 5.3 and Lemma 6. □
Proposition 2. GTD and GTD2 are true stochastic gradient algorithms with respect to the objective function $L(\theta, y)$ of the saddle-point problem (5.12) with $M = I$ and $M = C = \Phi^\top \Xi \Phi$ (the covariance matrix), respectively.

Proof. It is easy to see that the gradient updates of the saddle-point problem (5.12) (ascending in $y$ and descending in $\theta$) may be written as

\begin{align*}
y_{t+1} &= y_t + \alpha_t (b - A\theta_t - My_t), \\
\theta_{t+1} &= \theta_t + \alpha_t A^\top y_t.
\end{align*}

(5.14)

We denote $\hat{M} := 1$ (respectively $\tilde{M} := \tilde{C}$) for GTD (respectively GTD2). We may obtain the update rules of GTD and GTD2 by replacing $A$, $b$, and $C$ in (5.14) with their unbiased estimates $\hat{A}$, $\hat{b}$, and $\hat{C}$ from Eq. 5.5, which completes the proof. \qed

5.4 Finite-sample Analysis

In this section, we provide a finite-sample analysis for a revised version of the GTD/GTD2 algorithms. We first describe the revised GTD algorithms in Section 5.4.1 and then dedicate the rest of Section 5.4 to their sample analysis. Note that from now on we use the $M$ matrix (and its unbiased estimate $\hat{M}_t$) to have a unified analysis for GTD and GTD2 algorithms. As described earlier, $M$ is replaced by the identity matrix $I$ in GTD and by the co-variance matrix $C$ (and its unbiased estimate $\tilde{C}_t$) in GTD2.
5.4.1 The Revised GTD Algorithms

The revised GTD algorithms that we analyze in this chapter (see Algorithm 9) have three differences with the standard GTD algorithms of Eqs. 5.6 and 5.7 (and Eq. 5.14).

- We guarantee that the parameters $\theta$ and $y$ remain bounded by projecting them onto bounded convex feasible sets $\Theta$ and $Y$ defined in Assumption 13. In Algorithm 9, we denote by $\Pi_\Theta$ and $\Pi_Y$, the projection onto sets $\Theta$ and $Y$, respectively. This is standard in stochastic approximation algorithms and has been used in off-policy TD($\lambda$) [Yu, 2012] and actor-critic algorithms (e.g., Bhatnagar et al. 2009).

- After $n$ iterations ($n$ is the number of training samples in $D$), the algorithms return the weighted (by the step size) average of the parameters at all the $n$ iterations (see Eq. 5.16).

- The step-size $\alpha_t$ is selected as described in the proof of Proposition 3 in the supplementary material. Note that this fixed step size of $O(1/\sqrt{n})$ is required for the high-probability bound in Proposition 3 (see Nemirovski et al. 2009 for more details).

5.4.2 Assumptions

In this section, we make several assumptions on the MDP and basis functions that are used in our finite-sample analysis of the revised GTD algorithms. These assumptions are quite standard and are similar to those made in the prior work on GTD algorithms [Sutton et al., 2008, 2009; Maei, 2011] and those made in the analysis of SG algorithms [Nemirovski et al., 2009].
Algorithm 9 Revised GTD Algorithms

1: for $t = 1, \ldots, n$ do
2: Update parameters

\[ y_{t+1} = \Pi_Y \left( y_t + \alpha_t (\hat{b}_t - \hat{A}_t \hat{\theta}_t - \hat{M}_t y_t) \right) \]
\[ \theta_{t+1} = \Pi_\Theta \left( \theta_t + \alpha_t \hat{A}_t^\top y_t \right) \] (5.15)

3: end for
4: OUTPUT

\[ \bar{\theta}_n := \frac{\sum_{t=1}^n \alpha_t \theta_t}{\sum_{t=1}^n \alpha_t}, \quad \bar{y}_n := \frac{\sum_{t=1}^n \alpha_t y_t}{\sum_{t=1}^n \alpha_t} \] (5.16)

Assumption 13. (Feasibility Sets) We define the bounded closed convex sets $\Theta \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^d$ as the feasible sets in Algorithm 9. We further assume that the saddle-point $(\theta^*, y^*)$ of the optimization problem (5.12) belongs to $\Theta \times Y$. We also define $D_\theta := \left[ \max_{\theta \in \Theta} ||\theta||_2^2 - \min_{\theta \in \Theta} ||\theta||_2^2 \right]^{1/2}$, $D_y := \left[ \max_{y \in Y} ||y||_2^2 - \min_{y \in Y} ||y||_2^2 \right]^{1/2}$, and $R = \max \left\{ \max_{\theta \in \Theta} ||\theta||_2, \max_{y \in Y} ||y||_2 \right\}$.

Assumption 14. (Non-singularity) We assume that the covariance matrix $C = \mathbb{E}[\phi_i \phi_i^\top]$ and matrix $A = \mathbb{E}[\rho_i \phi_i (\Delta \phi_i)^\top]$ are non-singular.

Assumption 15. (Boundedness) We assume the features $(\phi_i, \phi'_i)$ have uniformly bounded second moments. This together with the boundedness of features (by $L$) and importance weights (by $\rho_{\max}$) guarantees that the matrices $A$ and $C$, and vector $b$ are uniformly bounded.

This assumption guarantees that for any $(\theta, y) \in \Theta \times Y$, the unbiased estimators of $b - A\theta - My$ and $A^\top y$, i.e.,
\[
\mathbb{E}[\hat{b}_t - \hat{A}_t \theta - \hat{M}_t y] = b - A \theta - My,
\]
\[
\mathbb{E}[\hat{A}_t^\top y] = A^\top y,
\]
\[(5.17)\]
all have bounded variance, i.e.,
\[
\mathbb{E}[||\hat{b}_t - \hat{A}_t \theta - \hat{M}_t y - (b - A \theta - My)||^2] \leq \sigma_1^2;
\]
\[
\mathbb{E}[||\hat{A}_t^\top y - A^\top y||^2] \leq \sigma_2^2,
\]
\[(5.18)\]
where \(\sigma_1\) and \(\sigma_2\) are non-negative constants. We further define
\[
\sigma^2 = \sigma_1^2 + \sigma_2^2.
\]
\[(5.19)\]
Assumption 15 also gives us the following “light-tail” assumption. There exist constants \(M_{*, \theta}\) and \(M_{*, y}\) such that
\[
\mathbb{E}[\exp\{||\hat{b}_t - \hat{A}_t \theta - \hat{M}_t y||^2_{M_{*, \theta}}\}] \leq \exp\{1\},
\]
\[
\mathbb{E}[\exp\{||\hat{A}_t^\top y||^2_{M_{*, y}}\}] \leq \exp\{1\}.
\]
\[(5.20)\]
This “light-tail” assumption is equivalent to the assumption in Eq. 3.16 in [Nemirovski et al., 2009] and is necessary for the high-probability bound of Proposition 3. We show how to compute \(M_{*, \theta}, M_{*, y}\) in the Appendix.
5.4.3 Finite-Sample Performance Bounds

The finite-sample performance bounds that we derive for the GTD algorithms in this section are for the case that the training set $\mathcal{D}$ has been generated as discussed in Section 5.2. We further discriminate between the on-policy ($\pi = \pi_b$) and off-policy ($\pi \neq \pi_b$) scenarios. The sampling scheme used to generate $\mathcal{D}$, in which the first state of each tuple, $s_i$, is an i.i.d. sample from a distribution $\xi$, also considered in the original GTD and GTD2 papers is for the analysis of these algorithms, and not used in the experiments [Sutton et al., 2008, 2009]. Another scenario that can motivate this sampling scheme is when we are given a set of high-dimensional data generated either in an on-policy or off-policy manner, and $d$ is so large that the value function of the target policy cannot be computed using a least-squares method (that involves matrix inversion), and iterative techniques similar to GTD/GTD2 are required.

We first derive a high-probability bound on the error function of the saddle-point problem (5.12) at the GTD solution $(\bar{\theta}_n, \bar{y}_n)$. Before stating this result in Proposition 3, we report the following lemma that is used in its proof.

**Lemma 7.** The induced $\ell_2$-norm of matrix $A$ and the $\ell_2$-norm of vector $b$ are bounded by

$$||A||_2 \leq (1 + \gamma)\rho_{\text{max}} L^2 d, \quad ||b||_2 \leq \rho_{\text{max}} LR_{\text{max}}.$$  \hspace{1cm} (5.21)

**Proof.** See the supplementary material in the Appendix. \hfill $\Box$

**Proposition 3.** Let $(\bar{\theta}_n, \bar{y}_n)$ be the output of the GTD algorithm after $n$ iterations (see Eq. 5.16). Then, with probability at least $1 - \delta$, we have
Err(\bar{\theta}_n, \bar{y}_n) \leq \sqrt{\frac{5}{n} \left(8 + 2 \log \frac{2}{\delta}\right) R^2} \times \left(\rho_{\text{max}} L\left(2(1 + \gamma)Ld + \frac{R_{\text{max}}}{R}\right) + \tau + \frac{\sigma}{R}\right), \quad (5.22)

where \text{Err}(\bar{\theta}_n, \bar{y}_n) is the error function of the saddle-point problem (5.12) defined by Eq. 5.11, \(R\) is defined in Assumption 13, \(\sigma\) is from Eq. 5.19, and \(\tau = \sigma_{\text{max}}(M)\) is the largest singular value of \(M\), which means \(\tau = 1\) for GTD and \(\tau = \sigma_{\text{max}}(C)\) for GTD2.

**Proof.** See the supplementary material. \(\square\)

**Theorem 1.** Let \(\bar{\theta}_n\) be the output of the GTD algorithm after \(n\) iterations (see Eq. 5.16). Then, with probability at least \(1 - \delta\), we have

\[
\frac{1}{2} \|A\bar{\theta}_n - b\|^2 \leq \tau \xi_{\text{max}} \text{Err}(\bar{\theta}_n, \bar{y}_n). \quad (5.23)
\]

**Proof.** From Proposition 1, for any \(\theta\), we have

\[
\max_y L(\theta, y) = \frac{1}{2} \|A\theta - b\|^2_{M^{-1}}.
\]

Given Assumption 14, the system of linear equations \(A\theta = b\) has a solution \(\theta^*\), i.e., the (off-policy) fixed-point \(\theta^*\) exists, and thus, we may write

\[
\min_\theta \max_y L(\theta, y) = \min_\theta \frac{1}{2} \|A\theta - b\|^2_{M^{-1}} = \frac{1}{2} \|A\theta^* - b\|^2_{M^{-1}} = 0. \quad (5.25)
\]
In this case, we also have\(^{2}\)

\[
\min_\theta L(\theta, y) \leq \max_y \min_\theta L(\theta, y) \leq \min_\theta \max_y L(\theta, y) = \frac{1}{2} \| A\theta^* - b \|_M^2 = 0.
\]

(5.26)

From Eq. 5.26, for any \((\theta, y) \in \Theta \times Y\) including \((\bar{\theta}_n, \bar{y}_n)\), we may write

\[
\text{Err}(\bar{\theta}_n, \bar{y}_n) = \max_y L(\bar{\theta}_n, y) - \min_\theta L(\theta, \bar{y}_n) 
\geq \max_y L(\bar{\theta}_n, y) = \frac{1}{2} \| A\bar{\theta}_n - b \|_M^2.
\]

(5.27)

Since \(\| A\bar{\theta}_n - b \|_M^2 \leq \tau \xi_{\text{max}} \| A\bar{\theta}_n - b \|_M^2\), where \(\tau\) is the largest singular value of \(M\), we have

\[
\frac{1}{2} \| A\bar{\theta}_n - b \|_M^2 \leq \tau \xi_{\text{max}} \| A\bar{\theta}_n - b \|_M^2 \leq \tau \xi_{\text{max}} \text{Err}(\bar{\theta}_n, \bar{y}_n).
\]

(5.28)

The proof follows by combining Eqs. 5.28 and Proposition 3. It completes the proof.

With the results of Proposition 3 and Theorem 1, we are now ready to derive finite-sample bounds on the performance of GTD/GTD2 in both on-policy and off-policy settings.

5.4.3.1 On-Policy Performance Bound

In this section, we consider the on-policy setting in which the behavior and target policies are equal, i.e., \(\pi_b = \pi\), and the sampling distribution \(\xi\) is the stationary distribution.

\(^{2}\)We may write the second inequality as an equality for our saddle-point problem defined by Eq. 5.12.
distribution of the target policy $\pi$ (and the behavior policy $\pi_b$). We use Lemma 8 to derive our on-policy bound. The proof of this lemma can be found in [Geist et al., 2012].

**Lemma 8.** For any parameter vector $\theta$ and corresponding $\hat{v} = \Phi \theta$, the following equality holds

$$V - \hat{v} = (I - \gamma \Pi P)^{-1} [(V - \Pi V) + \Phi C^{-1} (b - A \theta)] .$$

(5.29)

Using Lemma 8, we derive the following performance bound for GTD/GTD2 in the on-policy setting.

**Proposition 4.** Let $V$ be the value of the target policy and $\bar{v}_n = \Phi \bar{\theta}_n$, where $\bar{\theta}_n$ defined by (5.16), be the value function returned by on-policy GTD/GTD2. Then, with probability at least $1 - \delta$, we have

$$||V - \bar{v}_n||_\Xi \leq \frac{1}{1 - \gamma} \left[ ||V - \Pi V||_\Xi + \frac{L}{\nu} \sqrt{2d \tau \xi_{\text{max}} \text{Err}(\bar{\theta}_n, \bar{y}_n)} \right]$$

(5.30)

where $\text{Err}(\bar{\theta}_n, \bar{y}_n)$ is upper-bounded by Eq. 5.22 in Proposition 3, with $\rho_{\text{max}} = 1$ (on-policy setting).

*Proof.* See the supplementary material.

**Remark:** It is important to note that Proposition 4 shows that the error in the performance of the GTD/GTD2 algorithm in the on-policy setting is of $O \left( \frac{L^2 \sqrt{\tau \xi_{\text{max}} \log \frac{1}{\delta}}}{\nu^{1/4}} \right)$.

Also note that the term $\frac{L}{\nu}$ in the GTD2 bound is the conditioning number of the covariance matrix $C$. 

93
5.4.3.2 Off-Policy Performance Bound

In this section, we consider the off-policy setting in which the behavior and target policies are different, i.e., $\pi_b \neq \pi$, and the sampling distribution $\xi$ is the stationary distribution of the behavior policy $\pi_b$. We assume that off-policy fixed-point solution exists, i.e., there exists a $\theta^*$ satisfying $A\theta^* = b$. Note that this is a direct consequence of Assumption 14 in which we assumed that the matrix $A$ in the off-policy setting is non-singular. We use Lemma 9 to derive our off-policy bound. The proof of this lemma can be found in [Kolter, 2011]. Note that $\kappa(D)$ in his proof is equal to $\sqrt{\rho_{\text{max}}}$ in this thesis.

Lemma 9. If $\Xi$ satisfies the following linear matrix inequality

\[
\begin{bmatrix}
\Phi^T \Xi \Phi & \Phi^T \Xi P \Phi \\
P \Phi^T \Xi \Phi & P \Phi^T \Xi \Phi
\end{bmatrix} \succeq 0
\]

and let $\theta^*$ be the solution to $A\theta^* = b$, then

\[
||V - \Phi \theta^*||_\Xi \leq \frac{1 + \gamma \sqrt{\rho_{\text{max}}}}{1 - \gamma} ||V - \Pi V||_\Xi.
\] (5.32)

Note that the condition on $\Xi$ in Eq. 5.31 guarantees that the behavior and target policies are not too far away from each other. Using Lemma 9, we derive the following performance bound for GTD/GTD2 in the off-policy setting.

Proposition 5. Let $V$ be the value of the target policy and $\tilde{v}_n = \Phi \tilde{\theta}_n$, where $\tilde{\theta}_n$ is defined by (5.16), be the value function returned by off-policy GTD/GTD2. Also let
the sampling distribution $\Xi$ satisfy the condition in Eq. 5.31. Then, with probability at least $1 - \delta$, we have

$$
||V - \bar{v}_n||_\Xi \leq \frac{1 + \gamma \sqrt{\rho_{\text{max}}}}{1 - \gamma} ||V - \Pi V||_\Xi + \sqrt{\frac{2\tau_C\tau \xi_{\text{max}}}{\sigma_{\text{min}}(A^+M^{-1}A)}} \text{Err}(\bar{\theta}_n, \bar{y}_n),
$$

where $\tau_C = \sigma_{\text{max}}(C)$.

Proof. See the supplementary material.

5.4.4 Accelerated Algorithm

As discussed at the beginning of Section 5.3, this saddle-point formulation not only gives us the opportunity to use the techniques for the analysis of SG methods to derive finite-sample performance bounds for the GTD algorithms, as we show in Section 5.4, but it also allows us to use the powerful algorithms that have been recently developed to solve the SG problems and derive more efficient versions of GTD and GTD2. Stochastic Mirror-Prox (SMP) [Juditsky et al., 2008] is an “almost dimension-free” non-Euclidean extragradient method that deals with both smooth and non-smooth stochastic optimization problems (see Juditsky and Nemirovski 2011 and Bubeck 2014 for more details). Using SMP, we propose a new version of GTD/GTD2, called GTD-MP/GTD2-MP, with the following update formula:³

---
³For simplicity, we only describe Mirror-Prox GTD methods where the mirror map is identity, which can also be viewed as extragradient (EG) GTD methods. Mahadevan et al. [2014] gives a more detailed discussion of a broad range of mirror maps in RL.
Algorithm 10 GTD2-MP

1: for $t = 1, \ldots, n$ do
2:  Update parameters

$$\delta_t = r_t - \theta_t^T \Delta \phi_t$$
$$y_t^m = y_t + \alpha_t (\rho_t \delta_t - \phi_t^T y_t) \phi_t$$
$$\theta_t^m = \theta_t + \alpha_t \rho_t \Delta \phi_t (\phi_t^T y_t)$$
$$\delta_t^m = r_t - (\theta_t^m)^T \Delta \phi_t$$
$$y_{t+1} = y_t + \alpha_t (\rho_t \delta_t^m - \phi_t^T y_t^m) \phi_t$$
$$\theta_{t+1} = \theta_t + \alpha_t \rho_t \Delta \phi_t (\phi_t^T y_t^m)$$

3: end for
4: OUTPUT

$$\bar{\theta}_n := \frac{\sum_{t=1}^n \alpha_t \theta_t}{\sum_{t=1}^n \alpha_t}, \quad \bar{y}_n := \frac{\sum_{t=1}^n \alpha_t y_t}{\sum_{t=1}^n \alpha_t}$$

$$y_t^m = y_t + \alpha_t (\hat{b}_t - \hat{A}_t \theta_t - \hat{M}_t y_t), \quad \theta_t^m = \theta_t + \alpha_t \hat{A}_t^T y_t,$$  \hspace{1cm} (5.34)

$$y_{t+1} = y_t + \alpha_t (\hat{b}_t - \hat{A}_t \theta_t^m - \hat{M}_t y_t^m), \quad \theta_{t+1} = \theta_t + \alpha_t \hat{A}_t^T y_t^m. \hspace{1cm} (5.35)$$

After $T$ iterations, these algorithms return $\bar{\theta}_T := \frac{\sum_{t=1}^T \alpha_t \theta_t}{\sum_{t=1}^T \alpha_t}$ and $\bar{y}_T := \frac{\sum_{t=1}^T \alpha_t y_t}{\sum_{t=1}^T \alpha_t}$. The details of the algorithm is shown in Algorithm 10, and the experimental comparison study between GTD2 and GTD2-MP is reported in Section 5.7.
5.5 Further Analysis

5.5.1 Acceleration Analysis

In this section, we discuss the convergence rate of the accelerated algorithms using off-the-shelf accelerated solvers for saddle-point problems. For simplicity, we discuss the error bound of \( \frac{1}{2}\|A\theta - b\|_{M^{-1}}^2 \), and the corresponding error bound of \( \frac{1}{2}\|A\theta - b\|_{\Xi}^2 \) and \( \|V - \bar{v}_n\|_{\Xi} \) can be likewise derived. As can be seen from the above analysis, the convergence rate of the GTD algorithms family is

\[
(GTD/GTD2) : O\left(\frac{\tau + \|A\|_2 + \sigma}{\sqrt{n}}\right)
\]  

(5.37)

In this section, we raise an interesting question: what is the “optimal” GTD algorithm? To answer this question, we review the convex-concave formulation of GTD2. According to convex programming complexity theory [Juditsky et al., 2008], the un-improvable convergence rate of the stochastic saddle-point problem (5.12) is

\[
(Optimal) : O\left(\frac{\tau + \|A\|_2 + \sigma}{n^2 + \frac{\|A\|_2}{n} + \frac{\sigma}{\sqrt{n}}}\right)
\]  

(5.38)

There are many readily available stochastic saddle-point solvers, such as the stochastic Mirror-Prox (SMP) [Juditsky et al., 2008] algorithm, which leads to our proposed GTD2-MP algorithm. SMP is able to accelerate the convergence rate of our gradient TD method to:

\[
(GTD2 - MP) : O\left(\frac{\tau + \|A\|_2}{n^2} + \frac{\sigma}{\sqrt{n}}\right).
\]  

(5.39)

The stochastic accelerated primal-dual (SAPD) method [Chen et al., 2013] can theoretically accelerate our GTD to the optimal convergence rate. (5.38). Due to space
limitations, we are unable to present a more complete description, and refer interested readers to Juditsky et al. [2008]; Chen et al. [2013] for more details.

5.5.2 Learning With Biased $\rho_t$

The importance weight factor $\rho_t$ is lower bounded by 0, but yet may have an arbitrarily large upper bound. In real applications, the importance weight factor $\rho_t$ may not be estimated exactly, i.e., the estimation $\hat{\rho}_t$ is a biased estimation of the true $\rho_t$. The stochastic gradient we obtained is not the unbiased gradient of $L(\theta, y)$ anymore. This falls into a broad category of learning with inexact stochastic gradient, or termed stochastic gradient methods with an inexact oracle [Devolder, 2011]. Given the inexact stochastic gradient, the convergence rate and performance bound become much worse than the results with exact stochastic gradient. Based on the analysis by Juditsky et al. [2008], we have the error bound for inexact estimation of $\rho_t$.

**Proposition 6.** Let $\bar{\theta}_n$ be defined as above. Assume at the $t$-th iteration, $\hat{\rho}_t$ is the estimation of the importance weight factor $\rho_t$ with bounded bias such that $\mathbb{E}[\hat{\rho}_t - \rho_t] \leq \epsilon$. The convergence rates of GTD/GTD2 algorithms with iterative averaging are as follows,

$$||A\bar{\theta}_n - b||_{M^{-1}}^2 \leq O \left( \frac{\tau + ||A||_2 + \sigma}{\sqrt{n}} \right) + O(\epsilon)$$

(5.40)

This implies that the inexact estimation of $\rho_t$ may cause disastrous estimation error, which implies that an exact estimation of $\rho_t$ is very important.
5.5.3 Finite-sample Analysis Of Online Learning

Another more challenging scenario is the online learning scenario, where the samples are interactively generated by the environment, or by an interactive agent. The difficulty lies in that the sample distribution does not follow i.i.d sampling condition anymore, but follows an underlying Markov chain $\mathcal{M}$. If the Markov chain $\mathcal{M}$’s mixing time is small enough, i.e., the sample distribution reduces to the stationary distribution of $\pi_b$ very fast, our analysis still applies. However, it is usually the case that the underlying Markov chain’s mixing time $\tau_{\text{mix}}$ is not small enough. The analysis result can be obtained by extending the result of recent work [Duchi et al., 2012] from strongly convex loss functions to saddle-point problems, which is non-trivial and is thus left for future work.

5.5.4 Discussion Of TDC Algorithm

Now we discuss the limitation of our analysis with regard to the temporal difference with correction (TDC) algorithm [Sutton et al., 2009]. Interestingly, the TDC algorithm seems not to have an explicit saddle-point representation, since it incorporates the information of the optimal $y_t^*(\theta_t)$ into the update of $\theta_t$, a quasi-stationary condition which is commonly used in two-time-scale stochastic approximation approaches. An intuitive answer to the advantage of TDC over GTD2 is that the TDC update of $\theta_t$ can be considered as incorporating the prior knowledge into the update rule: for a stationary $\theta_t$, if the optimal $y_t^*(\theta_t)$ has a closed-form solution or is easy to compute, then incorporating this $y_t^*(\theta_t)$ into the update law tends to accelerate the algorithm’s convergence performance. For the GTD2 update, note that there is a sum of two terms where $y_t$ appears, which are $\rho_t(\phi_t - \gamma \phi_t')(y_t^\top \phi_t) = \rho_t \phi_t(y_t^\top \phi_t) - \gamma \rho_t \phi_t'(y_t^\top \phi_t)$. 

99
Replacing $y_t$ in the first term with $y_t^* (\theta_t) = \mathbb{E} [\phi_t \phi_t^T]^{-1} \mathbb{E} [\rho_t \delta_t (\theta_t) \phi_t]$, we have the TDC update rule. Note that in contrast to GTD/GTD2, TDC is a two-time scale algorithm; Also, note that TDC does not minimize any objective functions and the convergence of TDC requires more restrictions than GTD2 as shown by Sutton et al. [2009].

5.6 Control Learning Extension

In this section, we discuss the control learning extension of the proximal gradient family algorithms. We first present a lemma bridging the connection between the forward-view and backward-view perspectives. Then based on the GQ [Maei and Sutton, 2010] algorithm, we propose the control learning extension of the GTD2-MP algorithm, which is termed the GQ-MP algorithm.

5.6.1 Extension to Eligibility Trace

The $T$ operator looks one-step ahead, but it would be beneficial to look multiple steps ahead [Sutton and Barto, 1998], which gives rise to the multiple-step Bellman operator $T^\lambda$, otherwise known as the $\lambda$-weighted Bellman operator [Sutton and Barto, 1998], which is an arithmetic mean of the power series of $T$, i.e.,

$$
T^\lambda = (1 - \lambda) \sum_{i=0}^{\infty} \lambda^i T^{i+1}, \lambda \in (0, 1)
$$

(5.41)

Correspondingly, the multiple-step TD error, also termed the $\lambda$-TD error $\delta^\lambda$ with respect to $\theta$ is defined as
\[
\mathbb{E}[\delta^\lambda(\theta)] = T^\lambda \hat{v} - \hat{v} = T^\lambda \Phi \theta - \Phi \theta
\] (5.42)

The objective function as in Eqn (5.3) is changed accordingly as follows by replacing \(T\) with \(T^\lambda\),

\[
J(\theta) = \|\Phi^\top \Xi(T^\lambda \hat{v} - \hat{v})\|_2^{M-1} = \|\mathbb{E}[\rho_i \phi_i \delta^\lambda_i(\theta)]\|_2^{M-1}
\] (5.43)

This is called the *forward view* since it calls for looking multiple steps ahead, which is difficult to implement in practice. The *backward view* using eligibility traces is easy to implement. The eligibility trace is defined in a recursive way as

\[
e_0 = 0
\] (5.44)

\[
e_t = \rho_t \gamma \lambda e_{t-1} + \phi_t
\]

We introduce Theorem 2 to bridge the gap between the backward and forward view.

**Theorem 2.** Maei [2011]; Geist and Scherrer [2014] There is an equivalence between forward view and backward view such that

\[
\mathbb{E}[\phi_i \delta^\lambda_i(\theta)] = \mathbb{E}[e_i \delta_i(\theta)]
\] (5.45)

The details of the forward view and the backward view can be seen in [Sutton and Barto, 1998], Theorem 11 in [Maei, 2011], and Proposition 6 in [Geist and Scherrer, 2014]. A natural extension to Eqn (5.45) is by multiplying the importance ratio factor on both sides of the equality as follows,

\[
\mathbb{E}[\rho_i \phi_i \delta^\lambda_i(\theta)] = \mathbb{E}[\rho_i e_i \delta_i(\theta)]
\] (5.46)
**Algorithm 11 Greedy-GQ**

Initialize $e_t = 0$, starting from $s_0$.

1: repeat
2: Take $a_t$ according to $\pi_b$, and arrive at $s_{t+1}$
3: Compute $a_t^* = \arg \max_a \theta^T \phi(s_t, a)$. If $a_t = a_t^*$, then $\rho_t = \frac{1}{\pi_b(a_t|s_t)}$; otherwise $\rho_t = 0$.
4: Compute $\theta_{t+1}, y_{t+1}$ according to GQ-MP-LEARN Algorithm.
5: Choose action $a_t$, and get $s_{t+1}, r_{t+1}$
6: Set $t \leftarrow t + 1$
7: until $s_t$ is an absorbing state;
8: Compute $\bar{\theta}_t, \bar{y}_t$

**5.6.2 Greedy-GQ Algorithm**

With the help of Theorem 2, we can convert the objective formulation in (5.43) to

$$J(\theta) = ||\mathbb{E}[\rho_t e_t \delta_t(\theta)]||^2_{M^{-1}}$$  \hspace{1cm} (5.47)

The corresponding primal-dual formulation is

$$J(\theta) = \max_y \left( \langle \mathbb{E}[\rho_t e_t \delta_t(\theta)], y \rangle - \frac{1}{2} ||y||^2_M \right)$$  \hspace{1cm} (5.48)

And thus the new algorithm can be derived as

$$\theta_{t+1} = \theta_t + \alpha_t \rho_t \Delta \phi_t(e_t^T y_t)$$

$$y_{t+1} = y_t + \alpha_t (\rho_t \delta_t e_t - M_t y_t)$$  \hspace{1cm} (5.49)

Correspondingly, Greedy-GQ($\lambda$) with importance sampling can be derived. Here we present the Greedy-GQ($\lambda$) algorithm. Here is the GQ-MP-LEARN algorithm, which is the core step of Greedy-GQ algorithm.
Algorithm 12 GQ-MP-LEARN

\[ e_t = \gamma \rho_t e_{t-1} + \phi_t \]
\[ \delta_t = r_t + \theta_t^T \Delta \phi_t \]
\[ y_{t+1}^m = y_t + \alpha_t \left( \rho_t e_t \delta_t - (\phi_t^T y_t) \phi_t \right) \]
\[ \theta_t^m = \theta_t + \alpha_t \rho_t \Delta \phi_t (e_t^T y_t) \]
\[ \delta_t^m = r_t + \theta_t^{mT} \Delta \phi_t \]

5.7 Empirical Evaluation

In this section, we compare the previous GTD2 method with our proposed GTD2-MP method using various domains with regard to their value function approximation performance. It should be mentioned that since the major focus of this chapter is on policy evaluation, the comparative study focuses on value function approximation and thus comparisons on control learning performance are not reported in this chapter.

5.7.1 Baird Domain

The Baird example [Baird, 1995] is a well-known example to test the performance of off-policy convergent algorithms. Constant stepsizes \( \alpha = 0.005 \) for GTD2 and \( \alpha = 0.004 \) for GTD2-MP are chosen via comparison studies as in [Dann et al., 2014]. Figure 5.1 shows the MSPBE curve of GTD2, GTD2-MP of 8000 steps averaged over 200 runs. We can see that GTD2-MP gives a significant improvement over the GTD2 algorithm in which both the MSPBE and variance are substantially reduced.
5.7.2 50-State Chain Domain

The 50 state chain [Lagoudakis and Parr, 2003a] is a standard MDP domain. There are 50 discrete states \( \{s_i\}_{i=1}^{50} \) and two actions moving the agent left \( s_i \rightarrow s_{\max(i-1,1)} \) and right \( s_i \rightarrow s_{\min(i+1,50)} \). The actions succeed with probability 0.9; failed actions move the agent in the opposite direction. The discount factor is \( \gamma = 0.9 \). The agent receives a reward of +1 when in states \( s_{10} \) and \( s_{41} \). All other states have a reward of 0. In this experiment, we compare the performance of the value approximation with respect to different stepsizes \( \alpha = 0.0001, 0.001, 0.01, 0.1, 0.2, \cdots , 0.9 \) using the BEBF basis [Parr et al., 2007]. Figure 5.2 shows the value function approximation result where the cyan curve is the true value function, the red dashed curve is the GTD result, and the black curve is the GTD2-MP result. From the figure, one can see that GTD2-MP is much more robust with respect to stepsize choice than the GTD2 algorithm.

![Baird Star Example](image_url)

**Figure 5.1.** Off-Policy Convergence Comparison
5.7.3 Energy Management Domain

In this experiment we compare the performance of the algorithms on an energy management domain. The decision maker must decide how much energy to purchase or sell subject to stochastic prices. This problem is relevant in the context of utilities as well as in settings such as hybrid vehicles. The prices are generated from a Markov chain process. The amount of available storage is limited and degrades with use. The degradation process is based on the physical properties of lithium-ion batteries and discourages fully charging or discharging the battery. The energy arbitrage problem is closely related to the broad class of inventory management problems, with the storage level corresponding to the inventory. However, there are no known results describing the structure of optimal threshold policies in energy storage.

Note that since this is an off-policy evaluation problem, the formulated $A\theta = b$ does not have a solution, and thus the optimal MSPBE($\theta^*$) (respectively MSBE($\theta^*$)) do not reduce to 0. The result is averaged over 200 runs, and $\alpha = 0.001$ for both GTD2.
and GTD2-MP is chosen via comparison studies for each algorithm. As can be seen from Figure 5.3, GTD2-MP performs much better than GTD2 in the transient state. Then after reaching the steady state, as can be seen from Table 5.1, we can see that GTD2-MP reaches better steady state solution than the GTD algorithm. Based on the above empirical results and many other experiments we have conducted in other domains, we can conclude that GTD2-MP usually performs much better than the “vanilla” GTD2 algorithm.

Figure 5.3. Energy Management Example
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MSPBE</th>
<th>MSBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD</td>
<td>46.743</td>
<td>80.050</td>
</tr>
<tr>
<td>GTD</td>
<td>164.378</td>
<td>231.569</td>
</tr>
<tr>
<td>GTD2</td>
<td>77.139</td>
<td>111.19</td>
</tr>
<tr>
<td>GTD-MP</td>
<td>30.170</td>
<td>44.627</td>
</tr>
<tr>
<td>GTD2-MP</td>
<td>27.891</td>
<td>41.028</td>
</tr>
</tbody>
</table>

Table 5.1. Steady State Performance Comparison of Battery Management Domain

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MSPBE</th>
<th>MSBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD</td>
<td>0.0423</td>
<td>0.0547</td>
</tr>
<tr>
<td>GTD2</td>
<td>0.0244</td>
<td>0.0300</td>
</tr>
<tr>
<td>GTD2-MP</td>
<td>0.0238</td>
<td>0.0297</td>
</tr>
</tbody>
</table>

Table 5.2. Steady State Performance Comparison of Bicycle Domain

5.7.4 Bicycle Balancing and Riding Task

The bicycle balancing and riding domain [Randløv and Alstrøm, 1998] is a complicated domain. The goal is to learn to balance and ride a bicycle to a target position from the starting location.

To make a fair comparison, the parameter settings are identical to the parameter settings in [Lagoudakis and Parr, 2003b]. The samples are generated via random walk, after which, we compare the value function approximation results of TD, GTD2 and GTD2-MP algorithm. From the

From Figure 5.4, we can see that both the GTD2 and GTD2-MP algorithms reach a much better learning curve than the TD algorithm with significantly reduced variance. Besides, the GTD2 and GTD2-MP algorithms reach better steady-state solutions than the TD algorithm, as shown in Table 5.2.
Figure 5.4. Energy Management Example
5.7.5 Comparison with Other First-Order Policy Evaluation Algorithms

Here we give an experimental comparison between the gradient-based TD algorithms and the TD algorithm. Based on the experimental results shown above, we make the following empirical conclusions:

- Of all the gradient-based algorithms, GTD2-MP is the clear winner.
- For small and medium scale problems, TD is an ideal choice as it converges faster at the initial stage. On the other hand, GTD2-MP often reach better a steady state solution given more number of iterations.
- For large scale problems, GTD2-MP is the clear winner over the TD method with both reduced variance and better final solution, as shown in the bicycle and energy management domain.
- There exist some domains where $T$ operator is not differentiable, and thus only TD-based algorithms can be applied, such as the optimal stopping problem in [Choi and Van Roy, 2006].

Figure 5.5. Summary of Comparisons between TD and GTD algorithm family

5.8 Summary

In this chapter, we showed how gradient TD methods can be shown to be true stochastic gradient methods with respect to a saddle-point primal-dual objective function, which paved the way for the finite-sample analysis of off-policy convergent gradient-based TD algorithms such as GTD and GTD2. We presented both error bound and performance bound, which shows that the value function approximation bound
of the GTD algorithms family is $O\left(\frac{d}{n^{1/4}}\right)$. Furthermore, we proposed two revised algorithms, namely the projected GTD2 algorithm and the accelerated GTD2-MP algorithm. There are many interesting directions for future research. Our framework can be easily used to design regularized sparse gradient off-policy TD methods. One interesting direction is to investigate the convergence rate and performance bound for the TDC algorithm, which lacks a saddle-point formulation. The other is to explore tighter value function approximation bounds for off-policy learning.
CHAPTER 6

CONCLUSION AND FUTURE WORK

In the thesis, we presented a new family of stochastic gradient-based TD algorithms. This is the first time that a principled and systematic framework is proposed where convergence rate, finite-sample guarantee, regularization, and acceleration are provided. Our algorithms can be viewed as stochastic primal-dual gradient methods with respect to a wide range of projected Bellman error based objective functions, such as the mean-square projected Bellman error (MSPBE). Along this line of research, we studied several important aspects, including acceleration using mirror descent, regularization using proximal gradient method, and convergence rate analysis and sample complexity analysis.

Besides building a stochastic gradient TD learning framework and sample complexity analysis, the major contributions of the thesis also include providing the control learning extensions. Eligibility traces are essential to TD learning because they bridge the temporal gaps in cause and effect when the experience is limited. Thus, we also studied incorporating the eligibility traces into the proximal gradient TD learning framework.

It should also be noted that although this thesis focuses on designing and analyzing proximal gradient TD methods in the context of policy evaluation, all these ideas
and tools such as compound operator splitting, mirror descent, and extragradient can also be extended to various kinds of approximate dynamic programming methods, and other many problems in game theory. A more general way to evaluate the contribution of the thesis is that it provides an integrated way to problems that goes beyond convex optimization to monotone inclusion, such as variational inequality problems. It can also be envisioned as a principled way to deal with problems with Markov Decision Processes (MDPs), where the stringent biased sampling requirement can be circumvented based on a reformulation of the problem structure.

6.1 Future Work

There are several promising future research directions with our proposed proximal gradient TD learning framework. The first promising direction is to explore other compound operator splitting techniques other than primal-dual splitting. As we have shown in previous chapters, new algorithms can be designed if there exist methods that can split the operator so that the product of expectations can be avoided, and this operator splitting formulation does not have to be the primal-dual formulation. We have explored two primal-dual formulations, one is based on the convex conjugate function, and the other is based on dual norm representation. It would be interesting to see if there are any other compound operator splitting techniques that will lead to a family of new algorithms along with possibly faster convergence rate.

The second interesting direction is to explore the stochastic optimization RL framework in deep RL, such as deep Q-network (DQN) [Mnih et al., 2015]. Representation learning is a major challenge faced by machine learning and artificial intelligence. In
RL problems, the goal of representation learning is to automatically construct the representation of state that can be updated from the agent’s sequence of interactions with the environment, including the observations of the state of the environment and actions the agent takes.

Predictive state representations (PSRs) provide a promising approach to representation learning in partially observable RL problems. The basic assumption of PSR theory is that an agent that can effectively predict the future will be able to act effectively within its environment, for example, to maximize expected sum of long-term rewards. In the PSR framework, every state variable is not directly accessible but is indirectly observable, and the observation of the state represents a specific prediction about future observations as well. The agent updates its predictions after each interaction with its environment. It has been shown that by carefully choosing a small number of predictions, it is possible to provide a sufficient statistics for predicting all the future experience; in other words, the predictions can maintain all the useful information from the agent’s previous interactions [Singh et al., 2004].

Temporal-difference networks (TD networks) are a type of PSR that may ask compound predictions, not just about future observations, but about future state variables (i.e. predictions of predictions). This enables TD networks to operate at a more abstract level than traditional PSRs. TD network is a promising direction for representation learning that combines the two fundamental ideas from both predictive representations and recurrent neural networks. However, current temporal-difference networks suffer from a major drawback: the learning algorithm may diverge, even in simple environments, thus make it intractable to use in real applications. To overcome this drawback and motivated by the convergence guarantee of GTD family of
algorithms, Silver [2012] proposed the gradient temporal difference networks, which has asymptotic convergence guarantees using stochastic approximation approach. An interesting and promising direction is to apply the primal-dual saddle-point framework to GTD networks, which will enable a broad variety of new algorithms, and will also span a full spectrum between recurrent neural networks, the PSR framework, and stochastic optimization using primal-dual formulation and mirror descent.

Another interesting future direction is to explore proximal gradient TD algorithms with transfer RL. Given multiple different but related tasks, knowledge transfer is desirable and will help faster learning, less sample complexity, and better generalization ability. There are various types of transfer learning at different levels, such as instance level transfer, feature level transfer, and parameter level transfer. As we know, from transfer learning perspective, off-policy learning is instance level transfer learning. It would be interesting to see if other transfer RL problems can be formulated as saddle-point problems and if there is similar finite-sample analysis as well.

The fourth interesting future direction is to explore stochastic optimization framework in risk-sensitive RL, where the cost of mistakes is very high. Research along this direction can be roughly categorized into two categories, either by setting up cost-sensitive objective function, such as the Conditional value-at-risk (CVaR) approach [Chow and Ghavamzadeh, 2014], or by using high-probability based on concentration inequalities [Thomas et al., 2015], or by using robust optimization [Petrik and Subramanian, 2014].

Recently, Chow et al. [2015] points out the connections between robust optimization and CVaR approach. It would be challenging and promising to integrate stochastic
optimization and robust optimization [Ben-Tal and Nemirovski, 2002; Ben-Tal et al., 2009] into RL to deal with stochasticity and deterministic uncertainties.

Another interesting direction is to design new objective functions for TD learning. Since Bellman error is an expectation function (of the TD error), both MSPBE and NEU are (weighted) norm of expectations of the TD error. This is where the biased sampling problem comes from. It would be desirable if a new set of objectives can be designed, in which the biased sampling problem can be avoided.

Last but not the least, as mentioned in Chapter 5, the current finite-sample analysis is based on the i.i.d sample condition. This i.i.d sample condition is a common condition in off-line learning, which is quite common in similar off-line learning algorithm analysis. However, proximal gradient TD algorithms can also be applied to online learning scenario, which calls for online saddle-point analysis. In real applications, it is often more intriguing that the agent learns via real-time interaction with the environment. As mentioned earlier, this analysis can be carried out by a non-trivial extension of the result of recent work [Duchi et al., 2012], which is left for future work.
APPENDIX A

CONVERGENCE ANALYSIS OF SPARSE Q ALGORITHM

Definition 2 [Ghavamzadeh et al., 2011]: $\Pi_l$ is the $\ell_1$-regularized projection defined as: $\Pi_l y = \Phi \alpha$ such that $\alpha = \arg\min_w \|y - \Phi w\|^2 + \beta \|w\|_1$, which is a non-expansive mapping with respect to weighted $l_2$ norm induced by the on-policy sample distribution setting, as proven in [Ghavamzadeh et al., 2011]. Let the approximation error $f(y, \beta) = \|y - \Pi_l y\|^2$.

Definition 3 (Empirical $\ell_1$-regularized projection): $\hat{\Pi}_l$ is the empirical $\ell_1$-regularized projection with a specific $\ell_1$ regularization solver, and satisfies the non-expansive mapping property. It can be shown using a direct derivation that $\hat{\Pi}_l \Pi T$ is a $\gamma$-contraction mapping. Any unbiased $\ell_1$ solver which generates intermediate sparse solution before convergence, e.g., SMIDAS solver after $t$-th iteration, comprises an empirical $\ell_1$-regularized projection.

Theorem 1 The approximation error $\|V - \hat{V}\|$ of Algorithm 2 is bounded by (ignoring dependence on $\pi$ for simplicity):

$$\|V - \hat{V}\| \leq \frac{1}{1 - \gamma} \times \left(\|V - \Pi V\| + f(V, \beta) + (M - 1)P(0) + \|w^*\|_1^2 \frac{M}{\alpha \tau n}\right)$$

(A.1)
where $\hat{V}$ is the approximated value function after $n$-th iteration, i.e., $\hat{V} = \Phi \hat{w}_n$.

$M = \frac{2}{2-4\alpha_t(p-1)e}$, $\alpha_t$ is the stepsize, $P(0) = \frac{1}{n} \sum_{i=1}^{n} \|\Pi V(s_i)\|_2^2$, $s_i$ is the state of $i$-th sample, $e = \frac{d}{2}$, $d$ is the number of features, and finally, $w^*$ is $\ell_1$-regularized projection of $\Pi V$ such that $\Phi w^* = \Pi_{\ell_1} \Pi V$.

**Proof:** In the on-policy setting, the solution given by Algorithm 2 is the fixed point of $\hat{V} = \hat{\Pi}_{\ell_1} \Pi TV$ and the error decomposition is illustrated in Figure A.1. The error can be bounded by the triangle inequality

$$||V - \hat{V}|| = ||V - \Pi TV|| + ||\Pi TV - \hat{\Pi}_{\ell_1} \Pi TV|| + ||\hat{\Pi}_{\ell_1} \Pi TV - \hat{V}||$$  \hspace{1cm} (A.2)

Since $\hat{\Pi}_{\ell_1} \Pi T$ is a $\gamma$-contraction mapping, and $\hat{V} = \hat{\Pi}_{\ell_1} \Pi TV$, we have

$$||\hat{\Pi}_{\ell_1} \Pi TV - \hat{V}|| = ||\hat{\Pi}_{\ell_1} \Pi TV - \hat{\Pi}_{\ell_1} \Pi \hat{V}|| \leq \gamma ||V - \hat{V}||$$  \hspace{1cm} (A.3)

So we have

$$(1 - \gamma)||V - \hat{V}|| \leq ||V - \Pi TV|| + ||\Pi TV - \hat{\Pi}_{\ell_1} \Pi TV||$$

**Figure A.1.** Error Bound and Decomposition
\[ \| V - \Pi TV \| \text{ depends on the expressiveness of the basis } \Phi, \text{ where if } V \text{ lies in span}(\Phi), \text{ this error term is zero.} \]

\[ \| \Pi TV - \Pi_{\ell_1} \hat{\Pi} TV \| \text{ is further bounded by the triangle inequality} \]

\[ \| \Pi TV - \Pi_{\ell_1} \Pi TV \| \leq \| \Pi TV - \Pi_{\ell_1} \Pi TV \| + \| \Pi_{\ell_1} \Pi TV - \hat{\Pi}_{\ell_1} \Pi TV \| \]

where \( \| \Pi TV - \Pi_{\ell_1} \Pi TV \| \) is controlled by the sparsity parameter \( \beta \), i.e., \( f(\Pi TV, \beta) = \| \Pi TV - \Pi_{\ell_1} \Pi TV \| \), where \( \varepsilon = \| \hat{\Pi}_{\ell_1} \Pi TV - \Pi_{\ell_1} \Pi TV \| \) is the approximation error depending on the quality of the \( \ell_1 \) solver employed. In Algorithm 2, the \( \ell_1 \) solver is related to the SMIDAS \( \ell_1 \) regularized mirror-descent method for regression and classification [Shalev-Shwartz and Tewari, 2011]. Note that for a squared loss function \( L(\langle w, x_i \rangle, y_i) = \| \langle w, x_i \rangle - y_i \|_2^2 \), we have \( |L'|^2 \leq 4L \). Employing the result of Theorem 3 in [Shalev-Shwartz and Tewari, 2011], after the \( n \)-th iteration, the \( \ell_1 \) approximation error is bounded by

\[ \varepsilon \leq (M - 1) P(0) + |w^*|_2^2 \frac{M}{\alpha t} \frac{M}{\alpha t n}, M = \frac{2}{2 - 4\alpha t(p - 1)e} \]

By rearranging the terms and applying \( V = TV \), Equation (A.1) can be deduced.
APPENDIX B

PROOF OF LEMMA 7

Proof. From the boundedness of the features (by $L$) and the rewards (by $R_{\text{max}}$), we have

\[
\|A\|_2 = \|\mathbb{E}[\rho_t \phi_t \Delta \phi_t^\top]\|_2 \\
\leq \max_s \|\rho(s) \phi(s) (\Delta \phi(s))^\top\|_2 \\
\leq \rho_{\text{max}} \max_s \|\phi(s)\|_2 \max_s \|\phi(s) - \gamma \phi'(s)\|_2 \\
\leq \rho_{\text{max}} \max_s \|\phi(s)\|_2 \max_s (\|\phi(s)\|_2 + \gamma \|\phi'(s)\|_2) \\
\leq (1 + \gamma) \rho_{\text{max}} L^2 d.
\]

The second inequality is obtained by the consistent inequality of matrix norm, the third inequality comes from the triangular norm inequality, and the fourth inequality comes from the vector norm inequality $\|\phi(s)\|_2 \leq \|\phi(s)\|_\infty \sqrt{d} \leq L \sqrt{d}$. The bound on $\|b\|_2$ can be derived in a similar way as follows.
$||b||_2 = ||E[\rho_t \phi_t r_t]||_2$

$\leq \max_s ||\rho(s) \phi(s) r(s)||_2$

$\leq \rho_{\text{max}} \max_s ||\phi(s)||_2 \max_s ||r(s)||_2$

$\leq \rho_{\text{max}} L R_{\text{max}}.$

It completes the proof. \qed
APPENDIX C

PROOF OF PROPOSITION 3

Proof. The proof of Proposition 3 mainly relies on Proposition 3.2 in [Nemirovski et al., 2009]. We just need to map our convex-concave stochastic saddle-point problem in Eq. 5.12, i.e.,
\[
\min_{\theta \in \Theta} \max_{y \in Y} \left( L(\theta, y) = \langle b - A\theta, y \rangle - \frac{1}{2} ||y||_M^2 \right)
\]
to the one in Section 3 of Nemirovski et al. [2009] and show that it satisfies all the conditions necessary for their Proposition 3.2. Assumption 13 guarantees that our feasible sets \(\Theta\) and \(Y\) satisfy the conditions in [Nemirovski et al., 2009], as they are non-empty bounded closed convex subsets of \(\mathbb{R}^d\). We also see that our objective function \(L(\theta, y)\) is convex in \(\theta \in \Theta\) and concave in \(y \in Y\), and also Lipschitz continuous on \(\Theta \times Y\). It is known that in the above setting, our saddle-point problem in Eq. 5.12 is solvable, i.e., the corresponding primal and dual optimization problems: \(\min_{\theta \in \Theta} \left[ \max_{y \in Y} L(\theta, y) \right]\) and \(\max_{y \in Y} \left[ \min_{\theta \in \Theta} L(\theta, y) \right]\) are solvable with equal optimal values, denoted \(L^*\), and pairs \((\theta^*, y^*)\) of optimal solutions to the respective problems from the set of saddle points of \(L(\theta, y)\) on \(\Theta \times Y\).

For our problem, the stochastic sub-gradient vector \(G\) is defined as
\[
G(\theta, y) = \begin{bmatrix} G_{\theta}(\theta, y) \\ G_y(\theta, y) \end{bmatrix} = \begin{bmatrix} -\hat{A}_t^T y \\ -(\hat{b}_t - \hat{A}_t \hat{\theta} - \hat{M}_t y) \end{bmatrix}.
\]
This guarantees that the deterministic sub-gradient vector
\[
g(\theta, y) = \begin{bmatrix} g_\theta(\theta, y) \\ -g_y(\theta, y) \end{bmatrix} = \begin{bmatrix} \mathbb{E}[G_\theta(\theta, y)] \\ -\mathbb{E}[G_y(\theta, y)] \end{bmatrix}
\]
is well-defined, i.e., \( g_\theta(\theta, y) \in \partial_\theta L(\theta, y) \) and \( g_y(\theta, y) \in \partial_y L(\theta, y) \).

We also consider the Euclidean stochastic approximation (E-SA) setting in [Ne-mirovski et al., 2009] in which the distance generating functions \( \omega_\theta : \Theta \to \mathbb{R} \) and \( \omega_y : Y \to \mathbb{R} \) are simply defined as
\[
\omega_\theta = \frac{1}{2} ||\theta||_2^2, \quad \omega_y = \frac{1}{2} ||y||_2^2,
\]
modulo 1 with respect to \( ||\cdot||_2 \), and thus, \( \Theta^o = \Theta \) and \( Y^o = Y \) (see pp. 1581 and 1582 in Nemirovski et al. 2009). This allows us to equip the set \( Z = \Theta \times Y \) with the distance generating function
\[
\omega(z) = \frac{\omega_\theta(\theta)}{2D_\theta^2} + \frac{\omega_y(y)}{2D_y^2},
\]
where \( D_\theta \) and \( D_y \) defined in Assumption 13.

Now that we consider the Euclidean case and set the norms to \( \ell_2 \)-norm, we can compute upper-bounds on the expectation of the dual norm of the stochastic subgradients
\[
\mathbb{E} \left[ ||G_\theta(\theta, y)||_{*,\theta}^2 \right] \leq M_{*,\theta}^2, \quad \mathbb{E} \left[ ||G_y(\theta, y)||_{*,y}^2 \right] \leq M_{*,y}^2,
\]
where \( ||\cdot||_{*,\theta} \) and \( ||\cdot||_{*,y} \) are the dual norms in \( \Theta \) and \( Y \), respectively. Since we are in the Euclidean setting and use the \( \ell_2 \)-norm, the dual norms are also \( \ell_2 \)-norm, and thus, to compute \( M_{*,\theta} \), we need to upper-bound \( \mathbb{E} \left[ ||G_\theta(\theta, y)||_2^2 \right] \) and \( \mathbb{E} \left[ ||G_y(\theta, y)||_2^2 \right] \).
To bound these two quantities, we use the following equality that holds for any random variable $x$:

\[ \mathbb{E}[||x||^2] = \mathbb{E}[||x - \mu_x||^2] + ||\mu_x||^2, \]

where $\mu_x = \mathbb{E}[x]$. Here how we bound $\mathbb{E}[||G_\theta(\theta, y)||^2]$, 

\[
\mathbb{E}[||G_\theta(\theta, y)||^2] = \mathbb{E}[||\hat{A}_t^T y||^2] \\
= \mathbb{E}[||\hat{A}_t^T y - A^T y||^2] + ||A^T y||^2 \\
\leq \sigma_2^2 + (||A||_2||y||_2)^2 \\
\leq \sigma_2^2 + ||A||_2^2 R^2,
\]

where the first inequality is from the definition of $\sigma_3$ in Eq. 5.18 and the consistent inequality of the matrix norm, and the second inequality comes from the boundedness of the feasible sets in Assumption 13. Similarly we bound $\mathbb{E}[||G_y(\theta, y)||^2]$ as follows:

\[
\mathbb{E}[||G_y(\theta, y)||^2] = \mathbb{E}[||\hat{b}_t - \hat{A}_t \theta - \hat{M}_t y||^2] \\
= ||b - A\theta + My||^2 \\
+ \mathbb{E}[||\hat{b}_t - \hat{A}_t \theta - \hat{M}_t y - (b - A\theta - My)||^2] \\
\leq (||b||_2 + ||A||_2||\theta||_2 + \tau||y||_2)^2 + \sigma_1^2 \\
\leq (||b||_2 + (||A||_2 + \tau)R)^2 + \sigma_1^2,
\]

where these inequalities come from the definition of $\sigma_1$ in Eq. 5.18 and the boundedness of the feasible sets in Assumption 13. This means that in our case we can compute $M_{*,\theta}^2, M_{*,y}^2$ as

\[
M_{*,\theta}^2 = \sigma_2^2 + ||A||_2^2 R^2, \\
M_{*,y}^2 = (||b||_2 + (||A||_2 + \tau)R)^2 + \sigma_1^2,
\]

123
and as a result

\[ M^*_s = 2D^2_0 M^*_{s,\theta} + 2D^2_y M^*_{s,y} = 2R^2(M^*_{s,\theta} + M^*_{s,y}) \]
\[ = R^2 \left( \sigma^2 + ||A||_2^2 R^2 + (||b||_2 + (||A||_2 + \tau) R) \right) \]
\[ \leq \left( R^2 (2||A||_2 + \tau) + R(\sigma + ||b||_2) \right)^2, \]

where the inequality comes from the fact that \( \forall a, b, c \geq 0, a^2 + b^2 + c^2 \leq (a + b + c)^2 \).

Thus, we may write \( M_s \) as

\[ M_s = R^2 (2||A||_2 + \tau) + R(\sigma + ||b||_2). \quad (C.1) \]

Now we have all the pieces ready to apply Proposition 3.2 in [Nemirovski et al., 2009] and obtain a high-probability bound on \( \text{Err}(\bar{\theta}_n, \bar{y}_n) \), where \( \bar{\theta}_n \) and \( \bar{y}_n \) (see Eq. 5.16) are the outputs of the revised GTD algorithm in Algorithm 9. From Proposition 3.2 in [Nemirovski et al., 2009], if we set the step-size in Algorithm 9 (our revised GTD algorithm) to \( \alpha_t = \frac{2c}{M_s \sqrt{3n}} \), where \( c > 0 \) is a positive constant, \( M_s \) is defined by Eq. C.1, and \( n \) is the number of training samples in \( \mathcal{D} \), with probability of at least \( 1 - \delta \), we have

\[ \text{Err}(\bar{\theta}_n, \bar{y}_n) \leq \sqrt{\frac{5}{n}} (8 + 2 \log \frac{2}{\delta}) R^2 \left( 2||A||_2 + \tau + \frac{||b||_2 + \sigma}{R} \right). \quad (C.2) \]

Note that we obtain Eq. C.2 by setting \( c = 1 \) and the “light-tail” assumption in Eq. 5.20 guarantees that we satisfy the condition in Eq. 3.16 in [Nemirovski et al., 2009], which is necessary for the high-probability bound in their Proposition 3.2 to hold. The proof is complete by replacing \( ||A||_2 \) and \( ||b||_2 \) from Lemma 7. \( \square \)
APPENDIX D

PROOF OF PROPOSITION 4

Proof. From Lemma 8, we have

\[ V - \bar{v}_n = (I - \gamma \Pi P)^{-1} \times \]
\[ \left[ (V - \Pi V) + \Phi C^{-1}(b - A\bar{\theta}_n) \right]. \]

Applying \( \ell_2 \)-norm with respect to the distribution \( \xi \) to both sides of this equation, we obtain

\[
||V - \bar{v}_n||_\Xi \leq ||(I - \gamma \Pi P)^{-1}||_\Xi \times \]
\[
(D.1) \left( ||V - \Pi V||_\Xi + ||\Phi C^{-1}(b - A\bar{\theta}_n)||_\Xi \right).
\]

Since \( P \) is the kernel matrix of the target policy \( \pi \) and \( \Pi \) is the orthogonal projection with respect to \( \xi \), the stationary distribution of \( \pi \), we may write

\[
||(I - \gamma \Pi P)^{-1}||_\Xi \leq \frac{1}{1 - \gamma}.
\]
Moreover, we may upper-bound the term $||\Phi C^{-1}(b - A\bar{\theta}_n)||_\Xi$ in (D.1) using the following inequalities:

\[
||\Phi C^{-1}(b - A\bar{\theta}_n)||_\Xi \\
\leq ||\Phi C^{-1}(b - A\bar{\theta}_n)||_2 \sqrt{\xi_{\text{max}}} \\
\leq ||\Phi||_2 ||C^{-1}||_2 ||(b - A\bar{\theta}_n)||_{M^{-1}} \sqrt{\tau \xi_{\text{max}}} \\
\leq (L\sqrt{d})(\frac{1}{\nu}) \sqrt{2\text{Err}(\bar{\theta}_n, \bar{y}_n)} \sqrt{\tau \xi_{\text{max}}} \\
= \frac{L}{\nu} \sqrt{2d\tau \xi_{\text{max}} \text{Err}(\bar{\theta}_n, \bar{y}_n)},
\]

where the third inequality is the result of upper-bounding $||(b - A\bar{\theta}_n)||_{M^{-1}}$ using Eq. 5.28 and the fact that $\nu = 1/||C^{-1}||_2^2 = 1/\lambda_{\text{max}}(C^{-1}) = \lambda_{\text{min}}(C)$ ($\nu$ is the smallest eigenvalue of the covariance matrix $C$).
APPENDIX E

PROOF OF PROPOSITION 5

Proof. Using the triangle inequality, we may write

\[ ||V - \bar{v}_n||_\Xi \leq ||\bar{v}_n - \Phi \theta^*||_\Xi + ||V - \Phi \theta^*||_\Xi. \] (E.1)

The second term on the right-hand side of Eq. E.1 can be upper-bounded by Lemma 9. Now we upper-bound the first term as follows:

\[ ||\bar{v}_n - \Phi \theta^*||_\Xi^2 \]

\[ = ||\Phi \bar{\theta}_n - \Phi \theta^*||_\Xi^2 \]

\[ = ||\bar{\theta}_n - \theta^*||_C^2 \]

\[ \leq ||\bar{\theta}_n - \theta^*||_{A^T M^{-1} A}^2 ||(A^T M^{-1} A)^{-1}||_2 ||C||_2 \]

\[ = ||A(\bar{\theta}_n - \theta^*)||_{M^{-1}}^2 ||(A^T M^{-1} A)^{-1}||_2 ||C||_2 \]

\[ = ||A\bar{\theta}_n - b||_{M^{-1}}^2 \frac{\tau_C}{\sigma_{\min}(A^T M^{-1} A)}, \]

where \( \tau_C = \sigma_{\max}(C) \) is the largest singular value of \( C \), and \( \sigma_{\min}(A^T M^{-1} A) \) is the smallest singular value of \( A^T M^{-1} A \). Using the result of Theorem 1, with probability at least \( 1 - \delta \), we have
\[
\frac{1}{2} ||A\tilde{\theta}_n - b||_M^2 \leq \tau \xi_{\text{max}} \text{Err} (\tilde{\theta}_n, \bar{y}_n).
\] (E.2)

Thus,

\[
||v_n - \Phi \theta^*||_2^2 \leq \frac{2\tau C \tau \xi_{\text{max}}}{\sigma_{\text{min}}(A^+M^{-1}A)} \text{Err} (\tilde{\theta}_n, \bar{y}_n)
\] (E.3)

From Eqs. E.1, 5.32, and E.3, the result of Eq. 5.33 can be derived, which completes the proof. \qed
The problem represents an energy arbitrage model with multiple finite known price levels and a stochastic evolution given a limited storage capacity. In particular, the storage is assumed to be an electrical battery that degrades when energy is stored or retrieved. Energy prices are governed by a Markov process with states Θ. There are two energy prices in each time step: \( p^i : \Theta \to \mathbb{R} \) is the purchase (or input) price and \( p^o : \Theta \to \mathbb{R} \) is the sell (or output) price. Energy prices \( \theta \) vary between 0 and 10 and their evolution is governed by a martingale with a normal distribution around the mean.

We use \( s \) to denote the available battery capacity with \( s_0 \) denoting the initial capacity. The current state of charge is denotes as \( x \) or \( y \) and must satisfy that \( 0 \leq x_t \leq s_t \) at any time step \( t \). The action is the amount of energy to charge or discharge, which is denoted by \( u \). Positive \( u \) indicates that energy is purchased to charge the battery; negative \( u \) indicates the sale of energy.

The battery storage degrades with use. The degradation is a function of the battery capacity when charged or discharged. We use a general model of battery degradation with a specific focus on Li-ion batteries. The degradation function \( d(x, u) \in \mathbb{R} \) represent the battery capacity loss after starting at the state of charge \( x \geq 0 \) and charging
(discharging if negative) by $u$ with $-x \leq u \leq s_0$. This function indicates the loss of capacity, such that:

$$s_{t+1} = s_t - d(x_t, u_t)$$

The state set in the Markov decision problem is composed of $(x, s, \theta)$ where $x$ is the state of charge, $s$ is the battery capacity, and $\theta \in \Theta$ is the state of the price process. The available actions in a state $(x, s, \theta)$ are $u$ such that $-x \leq u \leq s - x$. The transition is from $(x_t, s_t, \theta_t)$ to $(x_{t+1}, s_{t+1}, \theta_{t+1})$ given action $u_t$ is:

$$x_{t+1} = x_t + u_t$$

$$s_{t+1} = s_t - d(x_t, u_t)$$

The probability of this transition is given by $P[\theta_{t+1}|\theta_t]$. The reward for this transition is:

$$r((x_t, s_t, \theta_t), u_t) = \begin{cases} 
-u_t \cdot p^i - c^d \cdot d(x_t, u_t) & \text{if } u_t \geq 0 \\
-u_t \cdot p^o - c^d \cdot d(x_t, u_t) & \text{if } u_t < 0 
\end{cases}.$$ 

That is, the reward captures the monetary value of the transaction minus a penalty for degradation of the battery. Here, $c^d$ represents the cost of a unit of lost battery capacity.

The Bellman optimality equations for this problem are:
\( q_T(x, s, \theta) = 0 \)

\[ v_t(x, s, \theta_t) = \min \{ p_{\theta_1}^i[u]_+ + p_{\theta_1}^o[u]_- + c^d d(x, u) + q_t(x + u, s - d(x, u), \theta_t) : u \in [-x, s - x] \} \]

\( q_t(x, s, \theta_t) = \lambda \cdot E[v_{t+1}(x, s, \theta_{t+1})] \)

where \([u]_+ = \max\{u, 0\}\) and \([u]_- = \min\{u, 0\}\) and the expectation is taken over \( P(\theta_{t+1}|\theta_t) \).

The value function is approximated using piece-wise linear features of three types \( \phi^1, \phi^2, \phi^3 \) defined as a function of the MDP state as follows:

\[
\begin{align*}
\phi^1_{w,q}(x, s, \theta) &= \begin{cases} 
[x - w]_+ & \text{if } \theta = q \\
0 & \text{otherwise}
\end{cases} \\
\phi^2_{w,q}(x, s, \theta) &= \begin{cases} 
[s - w]_+ & \text{if } \theta = q \\
0 & \text{otherwise}
\end{cases} \\
\phi^3_{w,q}(x, s, \theta) &= \begin{cases} 
[s + x - w]_+ & \text{if } \theta = q \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]

Here, \( w \in \{0, 0.1, \ldots, 0.9, 1\} \) and \( q \in \Theta \).

These features can be conveniently used to approximate a piecewise linear function.
APPENDIX G

NOTATION

All vectors are assumed to be column vectors, unless specified otherwise.

1: Vector of all ones of the size appropriate in the context.

0: Vector of all zeros of the size appropriate in the context.

θ: Primal weight

̂θ: Dual weight in dual space of mirror descent

k: Number of active features, i.e., the cardinality of active feature set

γ: Discount factor in [0, 1].

π: A policy in the given MDP.

φ(s): Features of state s. Represents a vector.

φi: Feature i as a vector for all states.

Φ: Approximation basis, represented as matrix.

ρ: Regularization parameter for regularization h(x).

µ: Strong convexity parameter.

I: Identity matrix of the size appropriate in the context.

T: Bellman operator.
$P$: Transition kernel for an MDP.

$P^\pi$: A matrix that represents the probability of transiting from the state defined by the row to the state defined by the column, such that $P^\pi(i, j) = P(s_i, \pi(s_i), s_j)$.

$R^\pi$: Vector of the rewards for action $a$, such that $R^\pi(i, j) = R(s_i, \pi(s_i))$.

$Q$: State-action value function, also known as Q-function.

$S_{\rho}(\cdot)$: Entrywise soft-thresholding operator, where $S_{\rho}(x)_i = \text{sign}(x_i) \max(x_i - \rho, 0) = \text{sign}(x_i)[x_i - \rho, 0]_+$. 


