AFRL-OSR-VA-TR-2013-0567

LEARNING REPRESENTATION AND CONTROL IN MARKOV DECISION PROCESSES

SRIDHAR MAHADEVAN

UNIVERSITY OF MASSACHUSETTS

10/21/2013
Final Report

AIR FORCE RESEARCH LABORATORY
AF OFFICE OF SCIENTIFIC RESEARCH (AFOSR)/RSL
ARLINGTON, VIRGINIA 22203
AIR FORCE MATERIEL COMMAND

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
17-10-2013	Final Report	01-August-2010 to 31-July-2013

4. TITLE AND SUBTITLE

Learning Representation and Control in Markov Decision Processes

5a. CONTRACT NUMBER
AIR FORCE FA9550-10-1-0383

5b. GRANT NUMBER
AIR FORCE FA9550-10-1-0383

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Professor Sridhar Mahadevan

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Massachusetts
Office of Grants and Contracts Administration
70 Butterfield Terrace
Amherst, MA 01003-9242

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of Scientific Research (AFOSR)
875 N. Randolph Street, Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Public

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research investigated algorithms for approximately solving Markov decision processes (MDPs), a widely used model of sequential decision making. Much past work on solving MDPs in adaptive dynamic programming and reinforcement learning has assumed representations, such as basis functions, are provided by a human expert. The research investigated a variety of approaches to automatic basis construction, including reward-sensitive and reward-invariant methods, diagonalization and dilation methods, as well as orthogonal and over-complete representations. A unifying perspective on the various basis construction methods emerges from showing they result from different power series expansions of value functions, including the Neumann series expansion, the Laurent series expansion, and the Schultz expansion. The research also develops new computational algorithms for learning sparse solutions to MDPs using convex optimization methods.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*

Award FA9550-10-1-0383: Learning Representation and Control in Markov Decision Processes: Final Report

Sridhar Mahadevan
Department of Computer Science
University of Massachusetts Amherst, MA 01003
Tel: (413) 545-3140
mahadeva@cs.umass.edu
October 17, 2013

Project Final Report

The research carried out under this grant investigated the automated design of problem-specific representations for approximately solving Markov decision processes (MDPs), a widely used model of sequential decision making. Much past work on solving MDPs in adaptive dynamic programming and reinforcement learning has assumed representations, such as basis functions, are provided by a human expert. This assumption makes it difficult to design agents that can solve novel collections of tasks. The research investigated a variety of approaches to automatic basis construction, including reward-sensitive and reward-invariant methods, diagonalization and dilation methods, as well as orthogonal and overcomplete representations. A unifying perspective on the various basis construction methods emerges from showing they result from different power series expansions of value functions, including the Neumann series expansion, the Laurent series expansion, and the Schultz expansion.

The research also develops new computational algorithms for learning to solve Markov decision processes with an overcomplete representation. The main idea is to use a L_1 regularized penalty function that promotes finding sparse representations of value functions. A key idea investigated in the research is the use of *mirror-descent* optimization methods to find sparse representations of value functions. A new regularized off-policy bradient temporal-difference (TD) method was explored, which combines off-policy convergence, robustness to irrelevant features, and scalability to large problems. Experimental results show promising performance.

Contents

Project Narrative

1 Introduction

The automated design of representations is a longstanding challenge in artificial intelligence and machine learning. The overall goal of the research carried out was to explore algorithmic methods of constructing representations for solving sequential decision making tasks. Research in sequential decision making combines work in a number of areas, including artificial intelligence (AI) [71], dynamic programming (DP) [9], operations research (OR) [69], as well as machine learning (ML) [11]. Markov decision processes (MDPs) [24, 69] have emerged as a core mathematical framework for modeling sequential decision making. MDPs model sequential decision making under uncertainty, where states are fully observable, and actions cause a stochastic transition among states in a fixed constant unit of time. Extensions of MDPs to semi-Markov decision processes (SMDPs) [25] have been used to model temporally extended actions and exploit constraints from hierarchical task decompositions [2]. The basic MDP model has also been extended to situations where states are hidden, leading to the partially observable MDP (POMDP) model [33, 76].

While traditional methods for solving MDPs require complete knowledge of the reward (or cost) function and transition model, there has been substantial research over the past two decades on solving MDPs using sample-based methods. In particular, approximate dynamic programming (ADP) [68], neuro-dynamic programming [8], and reinforcement learning (RL) [80] are closely related approaches for solving large MDPs. The primary strengths of these frameworks are the ability to learn without a model, crucial in large state spaces; the incremental nature of learning, using methods like *temporal-difference learning* (TD) [81]; and the use of modern function approximation architectures, from nonparametric kernel methods [73] to parametric approaches like neural networks [8] and graphical models [35, 63]. RL, NDP, and ADP have been been highly successful in solving some large MDPs, including backgammon [83], cellphone scheduling [8], elevator scheduling [80], helicopter control [58] humanoid robotics [66], multiagent vehicle scheduling [68], and Tetris [8].

Yet, despite these many successes, a fundamental limitation of many of these approaches is that they depend on a human designer choosing a set of *features* or *basis functions* mapping the original state space into a lower-dimensional (or higher-dimensional) discrete or continuous space. This proposal addresses the problem of automatically constructing intermediate problem-dependent representations of an MDP, which simplifies its solution as well as enables the solution of related problems. This proposal investigates a variety of different approaches to automatic basis generation, including reward-sensitive and reward-invariant methods, diagonalization vs. dilation methods, orthogonal vs. overcomplete bases, and "flat" vs. multiscale methods. A unifying perspective on the different basis construction methods emerges by showing that they correspond to different power series expansions of the discounted value function of a policy. For example, Krylov bases [64, 67, 61, 72] and proto-value functions [49] can be related to terms in the Neumann series expansion. Drazin bases [46] correspond to terms in the Laurent series expansion, which builds on the theory of generalized spectral inverses in Markov chains [14] and MDPs [69]. Finally, multiscale diffusion wavelet bases [18, 48] corresponds to rewriting the Neumman series using a product of dyadic powers using the Schultz expansion [10].

The research investigated a series of computational challenges involved in scaling automatic basis construction approaches to large MDPs. In the model-based setting, compact representations of bases will be constructed using factored dynamic Bayes net (DBN) models using decision diagrams to compactly represent transition matrices and reward functions [23]. Decision diagram representations of basis functions will be developed for hierarchical semi-Markov decision processes [2] and concurrent factored SMDPs [70]. A multiscale wavelet-based algorithm for computing and storing basis functions will be studied. Basis functions for large SMDPs will be constructed by exploiting task hierarchies and temporally extended actions. Sparse bases will be constructed using L_1-based regularization [29, 36]. Incremental algorithms for computing spectral bases in the model-free reinforcement learning setting will be developed. The extension of basis construction methods to POMDPs will be studied. A broad algorithmic framework for learning representation and control for solving MDPs will be investigated, where both the low-dimensional representation of an MDP as well as the approximately optimal policy are simultaneously learned.

2 Technical Background

2.1 Markov Decision Processes

A Markov decision process (MDP) models a decision maker (or "agent") who interacts with an external environment by taking actions [24, 69]. The sequential decision problem is formulated as a set of "states", each of which represents a particular situation in the decision problem. For example, a state may be a configuration of pieces in a game of backgammon or chess, the location of a robot, the number of jobs waiting in a queueing system and so on. Actions cause the environment to stochastically transition to a new state. The desirability of a state is determined by an immediate "reward" or "payoff". The goal of the agent is to choose actions such that it maximizes its long term payoffs. The transition from one state to the next is governed by a probability distribution that reflects the uncertainty in the outcome of decisions.

Formally, a discrete Markov decision process (MDP) $M = (S, A, P^a_{ss'}, R^a_{ss'})$ is defined by a finite set of discrete states S, a finite set of actions A, a transition model $P^a_{ss'}$ specifying the distribution over future states s' when an action a is performed in state s, and a corresponding reward model $R^a_{ss'}$ specifying a scalar cost or reward. In continuous Markov decision processes, the set of states $\subseteq \mathbb{R}^d$.

A *value function* is a mapping $S \to \mathbb{R}$ or equivalently (in discrete MDPs) a vector $\in \mathbb{R}^{|S|}$. Given a policy $\pi : S \to A$ mapping states to actions, its corresponding value function V^π specifies the expected long-term discounted sum of rewards received by the agent in any given state s when actions are chosen using the policy. The value function can be written as a *Neumann series* involving the one-step expected rewards R^π (a column vector of size $|S|$), and the transition matrix P^π:

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = (I + \gamma P^\pi + (\gamma P^\pi)^2 + \ldots) R^\pi$$

The Neumann expansion yields a series of basis vectors called Krylov bases, which can be used to approximately solve MDPs [72, 67, 64, 61]. We will also introduce another expansion of the value function in terms of spectral inverses below. Given any deterministic policy π, the operator T^π is defined as

$$T^\pi(V)(s) = R_{s\pi(s)} + \gamma \sum_{s' \in S} P^{\pi(s)}_{ss'} V(s').$$

Here $R_{s\pi(s)} = \sum_{s' \in s} P^{\pi(s)}_{ss'} R^{\pi(s)}_{ss'}$ is the expected immediate reward. It can be shown that the value function $V^\pi(s)$ is a fixed point of T^π, that is $T^\pi(V^\pi)(s) = V^\pi(s)$. The *Bellman error* of an approximate value function $\hat{V}$ is defined as $T^\pi(\hat{V}) - \hat{V}$, and plays a key role in both basis construction as well as approximation methods to solve MDPs. The optimal policy π^* and optimal value function V^{π^*} of an MDP are defined as:

$$V^{\pi^*}(s) \equiv V^*(s) \geq V^\pi(s) \;\; \forall \pi, s \in S.$$

The optimal policy π^* is not in general unique. However, any optimal policy π^* defines the same unique optimal value function. The optimal value function can be shown to be a fixed point of another operator T^*, defined as follows.

$$T^*(V)(s) = \max_a \left(R_{sa} + \gamma \sum_{s' \in S} P^a_{ss'} V(s') \right).$$

Action value functions are mappings from $S \times A \to \mathbb{R}$, and represent a convenient reformulation of the value function. The *optimal action value* $Q^*(x, a)$ is defined as the long-term value of the nonstationary policy performing a first, and then acting optimally according to V^*:

$$Q^*(s, a) \equiv E\left(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a \right)$$

where $V^*(s) = \max_a Q^*(s, a)$, r_{t+1} is the actual reward received at the next time step, and s_{t+1} is the state resulting from executing action a in state s_t. The corresponding Bellman equation for $Q^*(x, a)$ is given as

$$Q^*(s, a) = R_{sa} + \gamma \sum_{s'} P^s_{ss'} \max_{a'} Q^*(s', a') \tag{1}$$

One advantage of action value functions is that they can be used to directly estimate the optimal policy, where $\pi^*(s) \in \text{argmax}_a Q^*(s, a)$.

2.2 Approximation Methods for Solving MDPs

For large discrete or continuous MDPs, it is not possible to compute the value function exactly, but instead the best approximation within a family of parametric or non-parametric representations is sought [8]. We restrict our focus to linear architectures, where the approximation $\hat{V} = \Phi w$ lies within the linear span of a set of basis functions forming the columns of the basis matrix Φ. The basis function matrix Φ is an $|S| \times k$ matrix, where each column represents a basis function, and each row specifies the feature vector $\phi(s) = [\phi_1(s), \ldots, \phi_k(s)]^T$ mapping a particular state s to $\mathbb{R}^k$. Applying the Bellman operator T^π to $\hat{V} = \Phi w$ may result in a vector that lies outside the column space of Φ, and so requires a projection step to compute the next iterate. Approximation algorithms for solving MDPs can be categorized into two main types: *Bellman residual* algorithms, which attempt to minimize the Bellman error $\min_w \|T^\pi(\Phi w) - \Phi w\|^2_{\rho^\pi}$, and *fixed point* methods, which attempt to minimize the *projected* Bellman error $\min_w \|M^\pi_\Phi T^\pi(\Phi w) - \Phi w\|^2_{\rho^\pi}$ [54]. Here, $\|.\|^2_{\rho^\pi}$ refers to a norm defined by the invariant distribution ρ^π of the Markov chain induced by the policy π. M^π_Φ is a projection matrix onto the space spanned by the basis Φ. The least squares expression for $M^\pi_\phi = \Phi\Phi^\dagger$, where $\Phi^\dagger = (\Phi^T D_{\rho^\pi}\Phi)^{-1}\Phi^T D_{\rho^\pi}$ is the pseudo-inverse of Φ under the norm defined by ρ^π. D_{ρ^π} is a diagonal matrix whose entries are given by the stationary distribution ρ^π. The solution to least-squares fixed point projection is given by the following equation.

$$w_{FP} = A^{-1}_{FP} b_{FP}, \text{ where } A_{FP} = \left[\Phi^T D_{\rho^\pi}(I - \gamma P^\pi)\Phi\right], \text{ and } b_{FP} = \Phi^T D_{\rho^\pi} R^\pi \tag{2}$$

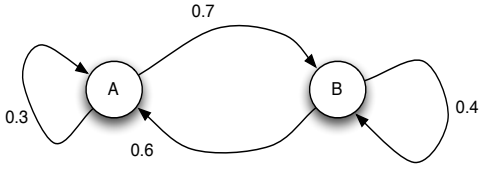
In the model-free reinforcement learning setting, the A matrix and b vector can be estimated from samples (s_t, a_t, r_t, s_{t+1}) generated from simulation. In particular, the update equations are:

$$\hat{A}_{FP} = \hat{A}_{FP} + \phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^T \qquad \hat{b}_{FP} = \hat{b}_{FP} + \phi(s_t) r_t.$$

The family of fixed-point algorithms includes least-squares TD (LSTD) [12], least-squares policy evaluation (LSPE) [55], as well as least-squares policy iteration (LSPI) [41]. LSPI uses a least-squares approach to approximate the action-value function $Q^\pi(s, a)$ for a policy π. In this case, basis functions $\phi(s, a)$ are defined over states and actions: $\hat{Q}^\pi(s, a; w) = \sum_{j=1}^k \phi_j(s, a) w_j$, where the w_j are weights or parameters that can be determined using a least-squares method. We have recently explored a new family of *hybrid least-squares* algorithms that interpolate between the fixed point approach and the Bellman residual approach [28]. Another important class of approximation methods for solving MDPs include linear programming (LP) based approaches, which also have traditionally used a fixed set of bases [19, 22].

2.3 Generalized Spectral Inverses in Markov Chains and MDPs

The discounted value function can also be expressed as a *Laurent series* expansion, which leads to another interesting type of basis. Given a transition matrix P, the low-rank singular matrix $L = I - P$ can be interpreted as a *generalized Laplacian* operator [1], since its row sums are 0 and its off-diagonal elements are non-positive. Figure 1 illustrates a simple example. A family of *spectral* inverses – the *Drazin inverse* and a special instance of it called the *group inverse* – have been widely studied in Markov chains [14] and MDPs [69].



$$\mathbb{L} = I - P = \begin{bmatrix} 0.7 & -0.7 \\ -0.6 & 0.6 \end{bmatrix}$$

Figure 1: The generalized Laplacian operator associated with a Markov chain.

The *index* of a square matrix $A \in \mathbb{C}^n \times \mathbb{C}^n$ is the smallest nonnegative integer k such that $\mathcal{R}(A^k) = \mathcal{R}(A^{k+1})$. Here, $\mathcal{R}(A)$ is the range (or column space) of matrix A. For example, a nonsingular (square) matrix A has index 0, because $\mathcal{R}(A^0) = \mathcal{R}(I) = \mathcal{R}(A)$. The Laplacian $\mathbb{L} = I - P$ of a Markov chain has

index 1. We can now define the Drazin inverse A^D of a square matrix A as the unique matrix satisfying the following properties:

$$A^D A A^D = A^D, \;\; A^D A = A A^D, \;\; A^{k+1} A^D = A^k, \tag{3}$$

where k is the index of A. When $k = 1$, the Drazin inverse becomes the group inverse, which satisfies the last property for the case $k = 1$, so $A^2 A^D = A$. It follows that $AA^D A = A$. For aperiodic chains, a simple interpretation of the group inverse of the Laplacian can be given based on the identities:

$$\mathbb{L}^D = (I - P)^D = (I - P + P^*)^{-1} - P^* = \lim_{n\to\infty} \sum_{k=0}^{n-1} \left(P^k - P^*\right).$$

In other words, $\mathbb{L}^D(i, j)$ in the group inverse matrix is the difference between the expected number of visits to state j starting in state i following the transition matrix P versus the expected number of visits to j following the long-term limiting matrix P^*.

The Drazin inverse is useful in linking average-reward and discounted MDPs [69]. Define the Laplacian matrix $\mathbb{L}^\pi = I - P^\pi$, where P^π is a transition matrix. We reformulate the Bellman equation in terms of an *interest rate* $\rho \equiv (1 - \gamma)\gamma^{-1}$ or equivalently, $\gamma = \frac{1}{1+\rho}$. The interpretation of ρ as an interest rate follows from the property that if a reward of 1 is "invested" at the first time step, then $1 + \rho$ is the amount received at the next time step. The interest rate formulation of the discounted value function associated with a policy π can be written as:

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = (1 + \rho)(\rho I + \mathbb{L}^\pi)^{-1} R^\pi$$

For $\rho > 0$, the matrix $(\rho I + \mathbb{L}^\pi)^{-1}$ is called the resolvent of $-\mathbb{L}^\pi$ at ρ. For $\gamma < 1$, the spectral radius of $\gamma P^\pi < 1$, and hence the inverse $(I - \gamma P^\pi)^{-1}$ exists. Consequently, the resolvent $(\rho I + \mathbb{L}^\pi)^{-1}$ also exists. Given a discounted MDP M and policy π, the value function V^π can be expressed in terms of the *gain* and *bias* of the associated Markov reward process $M_\pi = (P^\pi, R^\pi)$, and the Drazin inverse of the Laplacian, as follows [69]:

$$V^\pi = (1 + \rho) \left(\rho^{-1} g^\pi + h^\pi + \sum_{n=0}^{\infty} (-\rho)^n ((\mathbb{L}^\pi)^D)^{n+1} R^\pi \right), \tag{4}$$

where $\rho < \sigma(I - P^\pi)$ (the spectral radius of $I - P^\pi$). The *gain* $g^\pi = (P^\pi)^* R^\pi$ represents the expected reward (or cost) earned at each step. The *bias* represents the average-adjusted sum of rewards, namely $h^\pi = \sum_{t=0}^{\infty} ((P^\pi)^t - (P^\pi)^*) R^\pi$. As we show below, the successive terms of this expansion can be used as basis vectors, analogous to the use of Krylov bases from the Neumann expansion of a discounted value function.

3 Automatic Construction of Basis Functions

First, we describe a broad framework for constructing new features, or *basis functions*, for representing value functions. The framework shows how different approaches can be viewed as power-series expansions of the value function.

3.1 Basis Representations from Power Series Expansions of the Value Function

A fundamental question is how to automatically generate the basis matrix Φ used in approximation methods for solving MDPs. [1] We provide a unifying perspective in this section, showing several basis representations emerge naturally from different power series expansion of the value function. Figure 2 illustrates some of the dimensions along which the various construction methods can be categorized.

[1] We distinguish basis construction approaches from *basis adaptation* methods [53, 87] which search over a pre-defined collection of parametric bases, such as RBFs or polynomials.

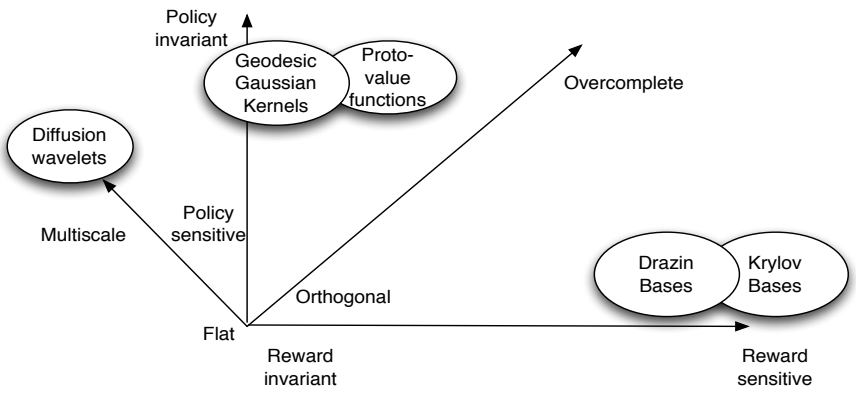


Figure 2: Different approaches to basis function construction can be categorized as shown here.

3.1.1 Spectral Bases

A widely used approach to dimensionality reduction in machine learning and statistics is to use eigenvectors of a suitable matrix, as in principal components analysis [31] and Laplacian eigenmaps [4]. A natural analog of this approach is to diagonalize the transition matrix P^π, and use its eigenvectors as a basis [64]. If the transition matrix P^π is reversible, there is a complete set of eigenvectors $\Phi^\pi = (\phi_1^\pi, \ldots \phi_n^\pi)$ that provides a change of basis in which the transition matrix P^π is representable as a diagonal matrix $P^\pi = \Phi^\pi \Lambda^\pi (\Phi^\pi)^T$, where Λ^π is a diagonal matrix of *eigenvalues*. Another way to express the above property is to write the transition matrix as a sum of *projection matrices* associated with each eigenvalue: $P^\pi = \sum_{i=1}^n \lambda_i^\pi \phi_i^\pi (\phi_i^\pi)^T$, where the eigenvectors ϕ_i^π form a complete orthogonal basis (i.e. $\parallel \phi_i^\pi \parallel_2 = 1$ and $\langle \phi_i^\pi, \phi_j^\pi \rangle = 0, i \neq j$). It readily follows that powers of P^π have the same eigenvectors, but the eigenvalues are raised to the corresponding power (i.e., $(P^\pi)^k \phi_i^\pi = (\lambda_i^\pi)^k \phi_i^\pi$). Since the basis matrix Φ spans all vectors on the state space S, we can express the reward vector R^π in terms of this basis as $R^\pi = \Phi^\pi \alpha^\pi$, where α^π is a vector of scalar weights. For high powers of the transition matrix, the projection matrices corresponding to the largest eigenvalues will dominate the expansion. Using the Neumann expansion of the value function, we get

$$V^\pi = \sum_{i=0}^{\infty} (\gamma P^\pi)^i \Phi^\pi \alpha^\pi = \sum_{i=0}^{\infty} \sum_{k=1}^{n} \gamma^i (P^\pi)^i \phi_k^\pi \alpha_k^\pi = \sum_{k=1}^{n} \sum_{i=0}^{\infty} \gamma^i (\lambda_k^\pi)^i \phi_k^\pi \alpha_k^\pi = \sum_{k=1}^{n} \frac{1}{1 - \gamma \lambda_k^\pi} \phi_k^\pi \alpha_k^\pi = \sum_{k=1}^{n} \beta_k \phi_k^\pi,$$

where we used the property that $(P^\pi)^i \phi_j^\pi = (\lambda_j^\pi)^i \phi_j^\pi$. Essentially, the value function V^π is represented as a linear combination of eigenvectors of the transition matrix. In order to provide the most efficient approximation, we can truncate the summation by choosing some small number $m < n$ of the eigenvectors, preferably those for whom β_k is large.

The spectral approach of diagonalizing the transition matrix is problematic since the transition matrix P^π cannot be assumed to be reversible, in which case one has to deal with complex eigenvalues (and eigenvectors). *Proto-value functions* (PVFs) [49, 26, 59] are an orthogonal reward-invariant basis using the eigenvectors of the symmetric graph Laplacian, which can be viewed as an "approximate" transition model that captures the topological properties of the underlying state space manifold. The graph is constructed by connecting adjacent states in an MDP reachable from each other under any policy. In a continuous MDP, the states are sampled by simulation, and a local distance metric such as Euclidean distance is used. The initial version was restricted to symmetric graphs on discrete MDPs [51], but later variants included continuous MDPs [50], directed graphs [26], and hierarchical SMDPs [59, 60]. We have recently extended this approach to partially observable MDPs (POMDPs) as well, where the graph is over high-dimensional belief states [84].

3.1.2 Krylov Bases

Another basis called the Krylov basis results from the Neumann expansion of the value function [64, 67, 72]. Krylov bases are generated by dilating the reward function by powers of the transition matrix. Given the

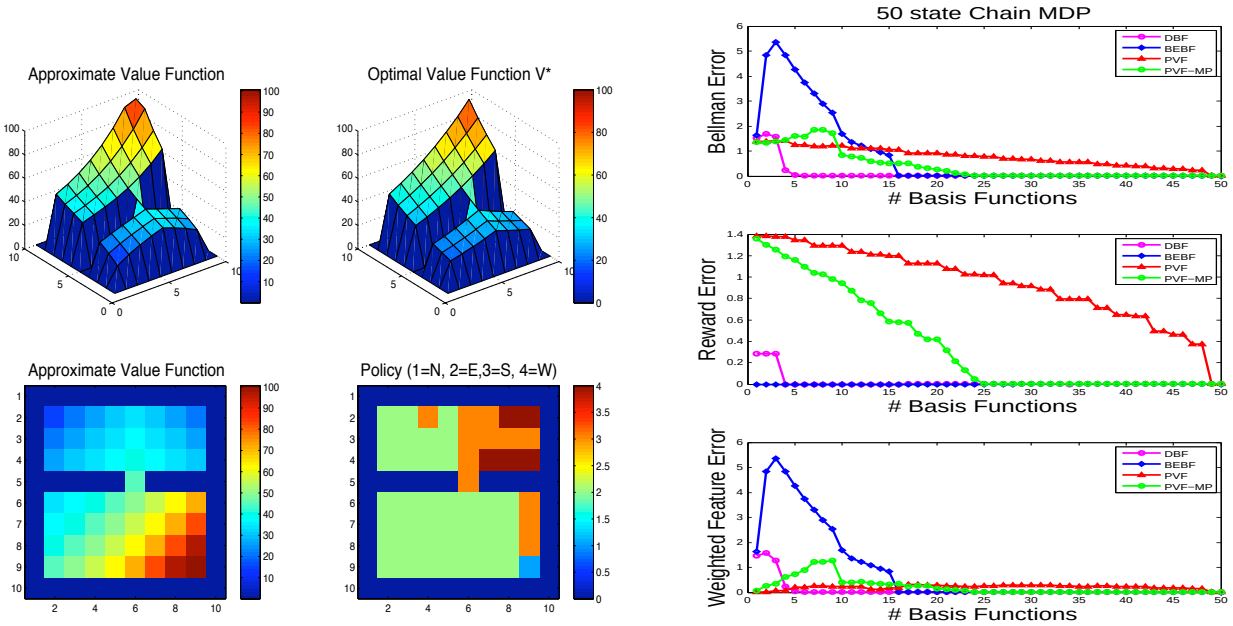


Figure 3: Left: A simple "two-room" robot navigation problem modeled as an MDP has 100 states. The agent is rewarded by 100 for reaching a corner goal state. Compass navigation actions are stochastic, succeeding with probability 0.9. Using just 4 Drazin bases, the optimal policy can be found. Right: Experimental comparison of several ways of automatically constructing bases on a 50 state chain MDP with rewards in state 10 and 41: DBF represents Drazin bases; BEBFs represent Bellman error bases; PVFs represent proto-value functions; PVF-MP represent a matching pursuit variant of PVFs. See Section 3.2 for an explanation of the plot.

transition matrix P^π and reward function R^π, the j^{th} *Krylov* subspace $\mathcal{K}_j$ is defined as the space spanned by the vectors:

$$\mathcal{K}^\pi{}_j = \{R^\pi, P^\pi R^\pi, (P^\pi)^2 R^\pi, \ldots, (P^\pi)^{j-1} R^\pi\} \tag{5}$$

Note that $\mathcal{K}^\pi{}_1 \subseteq \mathcal{K}^\pi{}_2 \subseteq \ldots$, such that for some $m, \mathcal{K}^\pi{}_m = \mathcal{K}^\pi{}_{m+1} = \mathcal{K}^\pi$. Thus, $\mathcal{K}^\pi$ is the P^π-invariant Krylov space generated by P^π and R^π. An incremental variant of the Krylov-based approach is called Bellman error basis functions (BEBFs), which is useful in the model-free reinforcement learning setting [34, 61]. As shown in Figure 3, Krylov (and BEBF bases can initially generate large approximation errors if the first few bases are highly localized around the region of a sparse reward.

3.1.3 Drazin Bases

The Laurent series expansion of the value function similarly motivates another basis construction method, which combines the Laplacian and Krylov-based approaches [46]. The *Drazin space* associated with a policy π and reward function R^π is defined as the space spanned by the set of Drazin vectors:

$$\mathbb{D}^\pi_m = \{P^* R^\pi, (\mathbb{L}^\pi)^D R^\pi, ((\mathbb{L}^\pi)^D)^2 R^\pi, \ldots, ((\mathbb{L}^\pi)^D)^{m-1} R^\pi\} \tag{6}$$

Intuitively, in dilating the reward function, we are not using the transition matrix P^π, but rather the modified transition matrix $P^\pi - P^*$. The first basis vector is the average-reward or gain $g^\pi = P^* R^\pi$ of policy π. The second basis vector is the product of $(\mathbb{L}^\pi)^D$ and R^π. Subsequent basis vectors are defined by the dilation of the reward function R^π by higher powers of the Drazin inverse $(\mathbb{L}^\pi)^D$ used in the Laurent series expansion of V^π. Figure 3 shows that Drazin bases outperforms the other bases on a two-room MDP. However, a drawback of Drazin bases is that they are computationally expensive to generate. A key challenge to be addressed in this research is developing incremental algorithms that approximate Drazin bases.

3.1.4 Multiscale Bases using the Schultz Expansion

Finally, we describe a multiscale policy-sensitive and overcomplete basis that emerges from another power series representation of the value function using the Schultz expansion [10]. This trick lets us rewrite the Neumann series as the product of dyadic powers (powers of two) of the discounted transition matrix γP^π:

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = \prod_{k=0}^{\infty} (I + (\gamma P^\pi)^{2^k}) R^\pi \tag{7}$$

In practice, the Schultz expansion converges far more quickly than the Neumann series. It is closely related to Successive Matrix Squaring (SMS) methods described below [15, 86]. However, computing the dyadic powers of the transition matrix seems a daunting prospect. In past work, we have shown how the multiscale diffusion wavelet algorithm [18] provides an efficient representation for dyadic powers of stochastic matrices [44] (see Figure 4).

The diffusion wavelet framework constructs a multi-resolution decomposition of the functions on a state space as a family of nested subspaces $V_0 \supseteq V_1 \supseteq \cdots \supseteq V_j \supseteq \ldots$. If the transition matrix P is interpreted as an operator on functions on the state space, then V_j is defined as the numerical range, up to precision ε, of $P^{2^{j+1}-1}$. P is assumed to be a sparse matrix, and that the numerical rank of the powers of P is assumed to decay rapidly for higher powers. A diffusion wavelet tree consists of orthogonal diffusion scaling functions Φ_j that are smooth bump functions, with some oscillations, at scale roughly 2^j, and orthogonal wavelets Ψ_j that are smooth localized oscillatory functions at the same scale. The scaling functions Φ_j span the subspace V_j, with the property that $V_{j+1} \subseteq V_j$, and the span of Ψ_j, W_j, is the orthogonal complement of V_j into V_{j+1}.

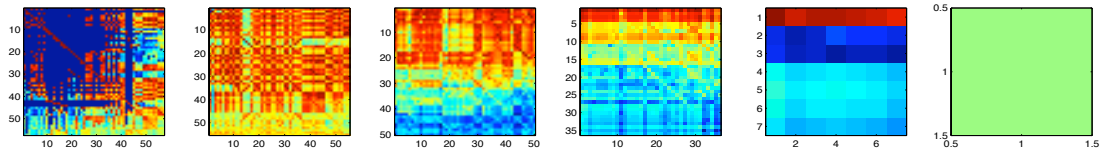


Figure 4: Compressed representation of dyadic powers of the transition matrix P^π in the two-room MDP (shown in Figure 3), from the first (extreme left) to the sixth (extreme right). Higher dyadic powers are compressed into progressively smaller sized arrays by representing the powers using a learned hierarchy of scaling function bases. For example, the sixth dyadic power ($2^6 = 64$) is effectively of size 1, a significant compression from its original 100×100 size. Entries are displayed in $\log_{10}$ scale.

A principal attraction of diffusion wavelet representations is their multiscale nature: they provide efficient ways of representing powers of stochastic matrices. One drawback of diffusion wavelets is that it can generate a large number of overcomplete bases, which needs to be effectively pruned using a basis selection algorithm. We discuss basis selection methods below. While the complexity of constructing a diffusion wavelet tree is quadratic in the size of the matrix, it can nevertheless be expensive to run the procedure on large stochastic matrices. We propose to address these challenges by developing a modified reward-sensitive diffusion wavelet procedure, which generates a sparse set of compact factored bases for a specific reward function, as well as to develop basis selection methods that exploit L_1 regularization.

3.2 Experimental Comparisons

An important objective of the research is to compare different methods for automatically constructing basis functions. Such comparisons can be done in several different settings: (i) Exact model-based experiments, which assume that the reward function and transition matrix of a fixed policy π are known (ii) Approximate model-free setting, where only sample trajectories and rewards are available. Figure 3 compares the performance of Drazin basis functions (DBFs), Bellman-error basis functions (BEBFs), and two variants of proto-value functions (PVFs and PVF-MP) on a 50 state chain MDP [41]. The two actions (go left, or go right) succeed with probability 0.9. When the actions fail, they result in movement in the opposite direction

with probability 0.1. The two ends of the chain are treated as "dead ends". Rewards of $+1$ are given in states 10 and 41. The PVF-MP algorithm selects basis functions incrementally based upon the Bellman error, where basis function k+1 is the PVF that has largest inner product with the Bellman error resulting from the previous k basis functions. The plots in the figure represent the following error terms, following the style of evaluation proposed in [62].

1. *Reward error:* The reward error measures the difference between the actual reward R^π and the approximated reward $R^\pi_\Phi = (\Phi^T\Phi)^{-1}\Phi^T R^\pi$ projected on the column space of bases Φ, which is plotted in the figure using the L_2 norm of the vector.

2. *Weighted feature error:* The weighted feature error is defined as the L_2 norm of the vector $\gamma\Delta_\Phi w_\Phi$, where $P_\Phi = (\Phi^T\Phi)^{-1}\Phi^T P^\pi\Phi$ is the projected transition matrix, and

$$\begin{aligned}\Delta_\Phi &= P\Phi - \Phi P_\Phi \\ w_\Phi &= \left(I - \gamma P^\pi_\Phi\right)^{-1} r_\Phi\end{aligned}$$

3. *Bellman error:* The Bellman error is the linear sum of the reward error and the feature error. [2]

PVFs can result in a large reward error if, as in this case, the reward function is poorly approximated by the eigenvectors of the combinatorial Laplacian on the chain graph. However, PVFs have very low weighted feature error. The overall Bellman error is largely due to the reward error. The reward error for BEBFs is by definition 0 as R^π is a basis vector itself. However, the feature error is extremely high, particularly initially, until around 15 bases are used. Consequently, the Bellman error for BEBFs remains high initially. Measured overall, Drazin bases have the best performance overall at this task. All three types of error are moderate initially and reduce quickly to 0 using around $5-6$ bases, three times as quickly as BEBFs and substantially faster than PVFs and PVF-MPs.

In addition to comparisons on simplified MDPs, the goal of the research is to also perform comparisons of basis construction methods on larger discrete and continuous MDPs, where exact models are unavailable or difficult to learn. A rigorous comparison of automatic basis generation methods would provide a valuable addition to the literature, and help practitioners who want to apply these methods to large MDPs.

3.3 Combining Different Bases

Since different basis construction methods often have complementary strengths and weaknesses, combining them may be an effective approach. We discuss two main ideas for integration of different bases.

3.3.1 Integrating Reward-Sensitive and Reward-Invariant Bases

One of the key goals of the research is to explore ways of combining reward-sensitive and reward-invariant bases. There are several reasons: reward-invariant bases are more transferrable and can be applied with many different reward functions on a state space; however, they may be less effective on specific reward functions and policies [62]. In semi-Markov decision processes, where the overall problem is hierarchically decomposed into a set of simpler SMDPs, lower level subproblems may be repeatedly called with slightly varying parameters (e.g., a common "navigation" routine that tasks a mobile robot to different rooms in a building). In this case, sharing basis functions among related subtasks seems useful. On the other hand, at higher levels, it may be more useful to use reward-sensitive bases.

Graph-theoretic bases, such as proto-value functions and geodesic Gaussian kernels [79], can be insensitive to the reward function if the graph is constructed without taking rewards into account. One approach to constructing reward-sensitive graph-theoretic bases is to incorporate rewards as part of the weight matrix in the graph-based Laplacian operator. The directed Laplacian operator [17] provides a convenient way of incorporating rewards into the graph representation, since it allows asymmetric edges. In a preliminary study, we found directed bases outperformed the non-directional bases in continuous MDPs [26] and

[2] The reward and weighted feature error vectors can individually be high, and yet cancel each other out. While the Bellman error is a natural metric to compare bases, other measures related to the quality of the learned policy also need to be used.

SMDPs [59]. Another approach is to construct "hybrid" bases. For example, Petrik [64] proposed combining "low-frequency" global Laplacian eigenvectors with "high-frequency" localized Krylov vectors, which helps alleviate the problem of large initial approximation errors.

4 Regularized Off-Policy Convergent Reinforcement Learning

In this section, we describe our recent work on regularized sparse methods for learning value functions. Our main ideas are to combine sophisticated methods of optimization, such as mirror descent, with work in reinforcement learning.

4.1 Regularized Off-Policy TD-Learning

We first present a novel l_1 regularized off-policy convergent TD-learning method (termed RO-TD), which is able to learn sparse representations of value functions with low computational complexity. The algorithmic framework underlying RO-TD integrates two key ideas: off-policy convergent gradient TD methods, such as TDC, and a convex-concave saddle-point formulation of non-smooth convex optimization, which enables first-order solvers and feature selection using online convex regularization. A detailed theoretical and experimental analysis of RO-TD is presented. A variety of experiments are presented to illustrate the off-policy convergence, sparse feature selection capability and low computational cost of the RO-TD algorithm.

Temporal-difference (TD) learning is a widely used method in reinforcement learning (RL). 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". Sutton et al. [82] introduced convergent off-policy temporal-difference learning 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 [85].

Regularizing reinforcement learning algorithms leads to more robust methods that can scale up to large problems with many potentially irrelevant features. LARS-TD [37] introduced a popular approach of combining l_1 regularization using Least Angle Regression (LARS) with the least-squares TD (LSTD) framework. Another approach was introduced in [30] (LCP-TD) based on the Linear Complementary Problem (LCP) formulation, an optimization approach between linear programming and quadratic programming. LCP-TD uses "warm-starts", which helps significantly reduce the burden of l_1 regularization. A theoretical analysis of l_1 regularization was given in [20], including error bound analysis with finite samples in the on-policy setting.

Another approach integrating Dantzig Selector with LSTD was proposed in [52], overcoming some of the drawbacks of LARS-TD. An approximate linear programming for finding l_1 regularized solutions of the Bellman equation was presented in [65]. All of these approaches are second-order methods, requiring complexity approximately cubic in the number of (active) features. Another approach to feature selection is to greedily add new features, proposed recently in [16]. Regularized first-order reinforcement learning approaches have recently been proposed primarily using the on-policy scenario. Here, the off-policy TD learning problem is formulated from the stochastic optimization perspective. A novel objective function is proposed based on the linear equation formulation of the 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 RO-TD algorithm is presented.

Here is a brief roadmap to the rest of the section. Section 4.2 reviews the basics of MDPs, RL and recent work on off-policy convergent TD methods, such as the TDC algorithm. Section 4.3 introduces the proximal gradient method and the convex-concave saddle point formulation of non smooth convex optimization. Section 4.5 presents the new RO-TD algorithm. Convergence analysis of RO-TD is presented in Section 4.10. Finally, in Section 4.11, experimental results are presented to demonstrate the effectiveness of RO-TD.

4.2 Off-Policy Convergent Methods for Reinfiorcement Learning

A *Markov Decision Process* (MDP) is defined by the tuple $(S, A, P^a_{ss'}, R, \gamma)$, comprised of a set of states S, a set of (possibly state-dependent) actions A (A_s), a dynamical system model comprised of the transition

kernel $P^a_{ss'}$ specifying the probability of transition to state s' from state s under action a, a reward model R, and $0 \leq \gamma < 1$ is a discount factor. A policy $\pi : S \to A$ is a deterministic mapping from states to actions. Associated with each policy π is a value function V^π, which is the fixed point of the Bellman equation:

$$V^\pi(s) = T^\pi V^\pi(s) = R^\pi(s) + \gamma P^\pi V^\pi(s)$$

where R^π is the expected immediate reward function (treated here as a column vector) and P^π is the state transition function under fixed policy π, and T^π is known as the *Bellman operator*. In what follows, we often drop the dependence of V^π, T^π, R^π on π, for notational simplicity. In linear value function approximation, a value function is assumed to lie in the linear span of a basis function matrix Φ of dimension $|S| \times d$, where d is the number of linear independent features. Hence, $V \approx \hat{V} = \Phi\theta$. The vector space of all value functions is a normed inner product space, where the "length" of any value function f is measured as $||f||^2_\Xi = \sum_s \xi(s) f^2(s) = f'\Xi f$ weighted by Ξ, where Ξ is defined in Figure 1. For the t-th sample, ϕ_t,ϕ'_t, θ_t and δ_t are defined in Figure 1. TD learning uses the following update rule $\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi_t$, where α_t is the stepsize. However, TD is only guaranteed to converge in the on-policy setting, although in many off-policy situations, it still has satisfactory performance [38]. TD with gradient correction (TDC) [82] aims to minimize the mean-square projected Bellman error (MSPBE) in order to guarantee off-policy convergence. MSPBE is defined as

$$\mathrm{MSPBE}(\theta) = \|\Phi\theta - \Pi T(\Phi\theta)\|^2_\Xi = (\Phi^T\Xi(T\Phi\theta - \Phi\theta))^T(\Phi^T\Xi\Phi)^{-1}\Phi^T\Xi(T\Phi\theta - \Phi\theta) \tag{8}$$

To avoid computing the inverse matrix $(\Phi^T\Xi\Phi)^{-1}$ and to avoid the double sampling problem [80] in (8), an auxiliary variable w is defined

$$w \quad = \quad (\Phi^T\Xi\Phi)^{-1}\Phi^T\Xi(T\Phi\theta - \Phi\theta) \tag{9}$$

The two time-scale gradient descent learning method TDC [82] is defined below

$$\theta_{t+1} = \theta_t + \alpha_t\delta_t\phi_t - \alpha_t\gamma\phi_t{}'(\phi_t^T w_t), w_{t+1} = w_t + \beta_t(\delta_t - \phi_t^T w_t)\phi_t \tag{10}$$

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

- Ξ 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 $A^{\frac{1}{2}}$. Note that $A^{\frac{1}{2}}$ may not be unique.
- $[\cdot,\cdot]$ is a row vector, and $[\cdot;\cdot]$ is a column vector.
- For the t-th sample, ϕ_t (the t-th row of Φ), ϕ'_t (the t-th row of Φ') are the feature vectors corresponding to s_t, s'_t, respectively. θ_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, $x_t = [w_t;\theta_t]$, α_t is a stepsize, $\beta_t = \eta\alpha_t, \eta > 0$.
- m, n are conjugate numbers if $\frac{1}{m} + \frac{1}{n} = 1, m > 0, n > 0$. $||x||_m = (\sum_j |x_j|^m)^{\frac{1}{m}}$ is the m-norm of vector x.
- ρ is l_1 regularization parameter, λ is the eligibility trace factor, N is the sample size, d is the number of basis functions, p is the number of active basis functions.

Figure 5: Notation used in this paper.

4.3 Proximal Gradient and Saddle-Point First-Order Algorithms

We now introduce some background material from convex optimization. The proximal mapping associated with a convex function h is defined as:[3]

$$prox_h(x) = \arg\min_u (h(u) + \frac{1}{2}\|u - x\|^2) \tag{11}$$

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:

$$prox_h(x)_i = S_\rho(x_i) = \max(x_i - \rho, 0) - \max(-x_i - \rho, 0) \tag{12}$$

where i is the index, and ρ is a threshold. With this background, we now introduce the proximal gradient method. If the optimization problem is

$$x^* = \arg\min_{x \in X} (f(x) + h(x)) \tag{13}$$

wherein $f(x)$ is a convex and differentiable loss function and the regularization term $h(x)$ is convex, possibly non-differentiable and computing $prox_h$ is not expensive, then computation of (13) can be carried out using the *proximal gradient method*:

$$x_{t+1} = prox_{\alpha_t h}(x_t - \alpha_t \nabla f(x_t)) \tag{14}$$

where $\alpha_t > 0$ is a (decaying) stepsize, a constant or it can be determined by line search.

4.4 Convex-concave Saddle-Point First Order Algorithms

The key novel contribution of our paper 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 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 (φ, 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 $SadVal = \inf_{x \in X} \sup_{y \in Y} \varphi(x, y)$ of φ 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 φ is often smooth and convex. The convex-concave saddle-point problems are, therefore, usually better suited for first-order methods [77]. A comprehensive overview on extending convex minimization to convex-concave saddle-point problems with unified variational inequalities is presented in [6].

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

$$f(x) := \|Ax - b\|_m = \max_{\|y\|_n < 1} y^T (Ax - b) \tag{15}$$

where m, n are conjugate numbers. Using the approach in [57], Equation (15) can be solved as

$$x_{t+1} = x_t - \alpha_t A^T y_t, y_{t+1} = \Pi_n(y_t + \alpha_t(Ax_t - b)) \tag{16}$$

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

$$\Pi_n y = \min(1, 1/\|y\|_n) y, n = 2, (\Pi_\infty y)_i = \min(1, 1/|y_i|) y_i, n = \infty \tag{17}$$

and Π_∞ is an entrywise operator.

4.5 Regularized off-policy convergent TD-Learning

We now describe a novel algorithm, regularized off-policy convergent TD-learning (RO-TD), which combines off-policy convergence and scalability to large feature spaces. The objective function is proposed based on the linear equation formulation of the TDC algorithm. Then the objective function is represented via its dual minimax problem. The RO-TD algorithm is proposed based on the primal-dual subgradient saddle-point algorithm, and inspired by related methods in [56],[57].

[3]The proximal mapping can be shown to be the resolvent of the subdifferential of the function h.

4.6 Objective Function of Off-policy TD Learning

In this subsection, we describe the objective function of the regularized off-policy RL problem. We now first formulate the two updates of θ_t, w_t into a single iteration by rearranging the two equations in (10) as $x_{t+1} = x_t - \alpha_t(A_t x_t - b_t)$, where $x_t = [w_t; \theta_t]$,

$$A_t = \begin{bmatrix} \eta\phi_t {\phi_t}^T & \eta\phi_t(\phi_t - \gamma\phi'_t)^T \\ \gamma\phi'_t {\phi_t}^T & \phi_t(\phi_t - \gamma\phi'_t)^T \end{bmatrix}, b_t = \begin{bmatrix} \eta r_t \phi_t \\ r_t \phi_t \end{bmatrix} \tag{18}$$

Following [82], the TDC algorithm solution follows from the linear equation $Ax = b$, where

$$A = \mathbb{E}[A_t(\phi_t, \phi'_t)], b = \mathbb{E}[b_t(\phi_t, r_t)], x = [w; \theta] \tag{19}$$

There are some issues regarding the objective function, which arise from the online convex optimization and reinforcement learning 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 objective 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 sample complexity in temporal difference learning: double-sampling. As pointed out in [80], 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), wherein the product of Bellman error or projected Bellman error metrics are involved. To overcome this sample complexity constraint, the product of TD errors should be avoided in the computation of gradients. To these ends, based on the linear equation formulation in (19) and the requirement on the objective function in above, we propose the regularized loss function as

$$L(x) = \min_{x \in X} \|Ax - b\|_m + h(x) \tag{20}$$

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., $L(\theta) = \mathrm{MSPBE}(\theta) + \rho\|\theta\|_1$. Because of the l_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 least square 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 $prox_{\alpha_t h(x_t)}(A_t x_t - b_t)$. 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[4].

4.7 RO-TD Algorithm Design

In this section, the problem of (20) is formulated as a convex-concave saddle-point problem, and the RO-TD algorithm is proposed. Analogous to (15), the regularized loss function can be formulated as

$$L(x) = \min_{x \in X} \|Ax - b\|_m + h(x) = \min_{x \in X} \max_{\|y\|_n \leq 1} (\langle y, Ax - b \rangle) + h(x) = L(x, y) \tag{21}$$

Similar to (16), Equation (21) can be solved via an iteration procedure as follows, where $x_t = [w_t; \theta_t]$.

$$\begin{aligned} x_{t+\frac{1}{2}} &= x_t - \alpha_t A_t^T y_t \quad , \quad y_{t+\frac{1}{2}} = y_t + \alpha_t(A_t x_t - b_t) \\ x_{t+1} &= prox_{\alpha_t h}(x_{t+\frac{1}{2}}) \quad , \quad y_{t+1} = \Pi_n(y_{t+\frac{1}{2}}) \end{aligned} \tag{22}$$

The averaging step, which plays a crucial role in stochastic optimization convergence, generates the *approximate saddle points*

$$\bar{x}_t = \left(\sum_{i=0}^t \alpha_i\right)^{-1} \sum_{i=0}^t \alpha_i x_i, \bar{y}_t = \left(\sum_{i=0}^t \alpha_i\right)^{-1} \sum_{i=0}^t \alpha_i y_i \tag{23}$$

[4]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.

Due to the computation of A_t in (22) at each iteration, the computation cost appears to be $O(Nd^2)$, where N is sample size defined in Figure 5. However, the computation cost is actually $O(Nd)$ with a linear algebraic trick by computing not A_t but $y_t^T A_t, A_t x_t - b_t$. Denoting $y_t = [y_{1,t}; y_{2,t}]$, where $y_{1,t}; y_{2,t}$ are column vectors of equal length, we have

$$y_t^T A_t = \left[\begin{array}{cc} \eta\phi_t^T(y_{1,t}^T\phi_t) + \gamma\phi_t^T(y_{2,t}^T\phi_t') & (\phi_t - \gamma\phi_t')^T(\eta y_{1,t}^T + y_{2,t}^T)\phi_t \end{array} \right] \tag{24}$$

$A_t x_t - b_t$ can be computed according to Equation (10) as follows:

$$A_t x_t - b_t = \left[\begin{array}{c} -\eta(\delta_t - \phi_t^T w_t)\phi_t; \gamma(\phi_t^T w_t)\phi_t{}' - \delta_t\phi_t \end{array} \right] \tag{25}$$

Both (24) and (25) are of linear computation complexity. Now we are ready to present the RO-TD algorithm:

Algorithm 1 RO-TD

Let π be some fixed policy of an MDP M, and let the sample set $S = \{s_i, r_i, s_i'\}_{i=1}^N$. Let Φ 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 $\langle y_t, A_t\rangle, A_t x_t - b_t$ in Equation (24) and (25).
4: Compute x_{t+1}, y_{t+1} as in Equation (22)
5: Set $t \leftarrow t + 1$;
6: **until** $t = N$;
7: Compute $\bar{x}_N, \bar{y}_N$ as in Equation (23) with $t = N$

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 $h(x) = \rho||x||_1$, $prox_{\alpha_t h}(\cdot) = S_{\alpha_t\rho}(\cdot)$. In real applications the sparsification requirement on θ 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 $S_{\alpha_t\rho}$ by two separate soft thresholding operations $S_{\alpha_t\rho_1}, S_{\alpha_t\rho_2}$ and thus the third equation in (22) is replaced by the following,

$$x_{t+\frac{1}{2}} = \left[w_{t+\frac{1}{2}}; \theta_{t+\frac{1}{2}}\right], \theta_{t+1} = S_{\alpha_t\rho_1}(\theta_{t+\frac{1}{2}}), w_{t+1} = S_{\alpha_t\rho_2}(w_{t+\frac{1}{2}}) \tag{26}$$

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

4.8 RO-GQ(λ) Design

GQ(λ)[43] 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 (10) to

$$\theta_{t+1} = \theta_t + \alpha_t[\delta_t e_t - \gamma(1-\lambda){w_t}^T e_t\bar{\phi}_{t+1}], w_{t+1} = w_t + \beta_t(\delta_t e_t - w_t^T\phi_t\phi_t) \tag{27}$$

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:

$$L(\theta) = ||\Phi\theta - \Pi T^{\pi\lambda}\Phi\theta||_\Xi^2 \tag{28}$$

$$A_t(\phi_t, \bar{\phi}_{t+1}) = \left[\begin{array}{cc} \eta\phi_t{\phi_t}^T & \eta e_t(\phi_t - \gamma\bar{\phi}_{t+1})^T \\ \gamma(1-\lambda)\bar{\phi}_{t+1}e_t^T & e_t(\phi_t - \gamma\bar{\phi}_{t+1})^T \end{array} \right], b_t(\phi_t, r_t) = \left[\begin{array}{c} \eta r_t e_t \\ r_t e_t \end{array} \right] \tag{29}$$

Similar to Equation (24) and (25), the computation of $\langle y_t, A_t\rangle, A_t x_t - b_t$ is

$$\begin{array}{rcl} \langle y_t, A_t\rangle & = & \left[\begin{array}{cc} \eta\phi_t^T(y_{1,t}^T\phi_t) + \gamma(1-\lambda)e_t^T(y_{2,t}^T\bar{\phi}_{t+1}) & (\phi_t - \gamma\bar{\phi}_{t+1})^T(\eta y_{1,t}^T + y_{2,t}^T)e_t \end{array} \right] \\ A_t x_t - b_t & = & \left[\begin{array}{c} -\eta(\delta_t e_t - \phi_t^T w_t\phi_t); \gamma(1-\lambda)(e_t^T w_t)\bar{\phi}_{t+1} - \delta_t e_t \end{array} \right] \end{array} \tag{30}$$

where eligibility traces e_t, and $\bar{\phi}_t, T^{\pi\lambda}$ are defined in [43]. Algorithm 2, RO-GQ(λ), extends the RO-TD algorithm to include eligibility traces.

Algorithm 2 RO-GQ(λ)

Let π and Φ be as defined in Algorithm 1. Starting from s_0.

1: **repeat**
2: Compute $\phi_t, \bar{\phi}_{t+1}$ and TD error $\delta_t = (r_t + \gamma \bar{\phi}_{t+1}^T \theta_t) - \phi_t^T \theta_t$
3: Compute $\langle y_t, A_t \rangle, A_t x_t - b_t$ in Equation (30).
4: Compute x_{t+1}, y_{t+1} as in Equation (22)
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 (23)

4.9 Extension

It is also noteworthy that to reach sparsity and regularization, there exists another formulation of loss function different from Equation (21) with the following convex-concave formulation as illustrated in [77]:

$$L(x) = \min_x \frac{1}{2} \|Ax - b\|_2^2 + \rho \|x\|_1 = \min_x \max_{\|A^T y\|_\infty \leq 1} (b^T y - \frac{\rho}{2} y^T y) = L(x, y) \tag{31}$$

This problem is able to be solved via FOM by adding an auxiliary variable v.

$$L(x, y, v) = \min_x \max_{\|y\|_\infty \leq 1, v} \left(x^T y + v^T (Ax - b) - \frac{\rho}{2} v^T v \right) \tag{32}$$

which can be solved iteratively without proximal gradient step as follows, which is a counterpart of Equation (22),

$$\begin{aligned} x_{t+1} = x_t - \alpha_t \rho (y_t + {A_t}^T v_t) \quad &, \quad v_{t+1} = v_t + \frac{\alpha_t}{\rho} (A_t x_t - b_t - \rho v_t) \\ y_{t+\frac{1}{2}} = y_t + \frac{\alpha_t}{\rho} x_t \quad &, \quad y_{t+1} = \Pi_\infty (y_{t+\frac{1}{2}}) \end{aligned} \tag{33}$$

Similar as the trick used in Equation (24), $v_t^T A_t$ is computed with linear computation cost by denoting $v_t = [v_{1,t}; v_{2,t}]$, where $v_{1,t}; v_{2,t}$ are column vectors of equal length, we have

$$v_t^T A_t = \left[\begin{array}{cc} \eta \phi_t^T (v_{1,t}^T \phi_t) + \gamma \phi_t^T (v_{2,t}^T \phi_t') & (\phi_t - \gamma \phi_t')^T (\eta v_{1,t}^T + v_{2,t}^T) \phi_t \end{array} \right] \tag{34}$$

4.10 Convergence Analysis of RO-TD

Assumption 1 (**MDP**)[82]: The underlying Markov Reward Process (MRP) $M = (S, P, R, \gamma)$ is finite and mixing, with stationary distribution π. Assume that $\exists$ a scalar $R_{\max}$ such that $Var[r_t | s_t] \leq R_{\max}$ holds w.p.1.
Assumption 2 (**Basis Function**): Φ is a full column rank matrix, namely, Φ comprises a linear independent set of basis functions w.r.t all sample states in sample set S. Also, assume the features (ϕ_t, ϕ_t') have uniformly bounded second moments. Finally, if (s_t, a_t, s_t') is an i.i.d sequence, $\forall t, \|\phi_t\|_\infty < +\infty, \|\phi_t'\|_\infty < +\infty$.
Assumption 3 (**Subgradient Boundedness**)[56]: Assume for the bilinear convex-concave loss function defined in (21), and the sets X, Y are closed compact sets. Then the subgradient $\langle y_t, A_t \rangle$ and $A_t x_t - b_t$ in RO-TD algorithm are uniformly bounded, i.e., there exists a constant L such that $\|A_t x_t - b_t\| \leq L, \|\langle y_t, A_t \rangle\| \leq L$.
Proposition 1: The approximate saddle-point $\bar{x}_t$ of RO-TD converges w.p.1 to the global minimizer of the following,

$$x^* = \arg\min_{x \in X} \|Ax - b\|_m + \rho \|x\|_1 \tag{35}$$

Proof Sketch: We give an outline of the proof here. We first present the monotone operator corresponding to the bilinear saddle-point problem and then extend it to the stochastic approximation case with certain restrictive assumptions, and use the result in [77].

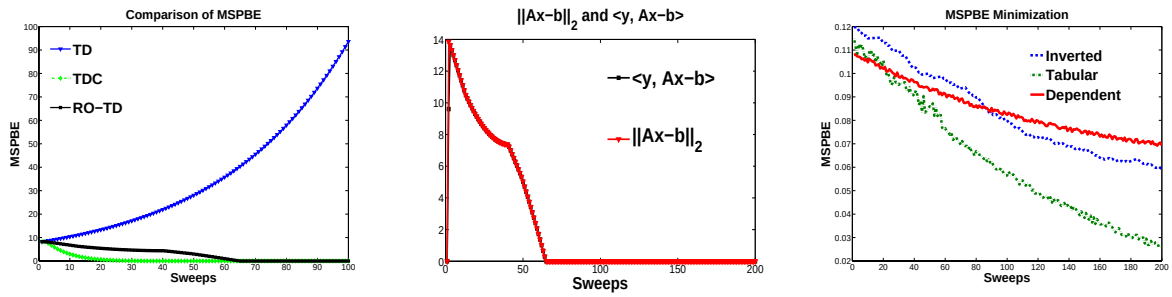


Figure 6: Illustrative examples of the convergence of RO-TD using the Star and Random-walk MDPs.

The monotone operator $\Phi(x, y)$ with saddle-point problem $SadVal = \inf_{x \in X} \sup_{y \in Y} \phi(x, y)$ is a point-to-set operator

$$\Phi(x, y) = \{\partial_x \phi(x, y)\} \times \{-\partial_y \phi(x, y)\}$$

where $\partial_x \phi(x, y)$ is the subgradient of $\phi(x, \cdot)$ over x and $\partial_y \phi(x, y)$ is the subgradient of $\phi(\cdot, y)$ over y. For the bilinear problem in Equation (15), the corresponding $\Phi(x, y)$ is

$$\Phi(x, y) = (A^T y, b - Ax) \tag{36}$$

Now we verify that the problem (20) can be reduced to a standard bilinear minimax problem. To prove this we only need to prove that the set X in our RL problem is indeed a closed compact convex set. This is easy to verify as we can simply define $X = \{x | \|x\|_2 \leq R\}$ where R is large enough. In fact, the sparse regularization $h(x) = \rho ||x||_1$ helps x_t stay within this l_2 ball. Now we extend this argument to the stochastic approximation case. Here, the objective function $f(x)$ is given by the stochastic oracle, and in our case, it is $Ax - b$, where A, b are defined in Equation (19). Given Assumption 3 and further assuming that the noise ε_t for t-th sample is i.i.d noise, then using the result in [32], we can prove that the RO-TD algorithm converges to the global minimizer of

$$x^* = \arg\min_{x \in X} \|Ax - b\|_m + \rho \|x\|_1$$

Then we prove the error level of approximate saddle-point $\bar{x}_t, \bar{y}_t$ defined in (23) is $\alpha_t L^2$. With the subgradient boundedness assumption and using the result in Proposition 1 in [56],

$$-\frac{1}{2\alpha t}\left(\|y_0 - y^*\|^2 + \|x_0 - \bar{x}_t\|^2\right) - \alpha L^2 \leq L(\bar{x}_t, \bar{y}_t) - L(x^*, y^*) \leq \frac{1}{2\alpha t}\left(\|y_0 - \bar{y}_t\|^2 + \|x_0 - x^*\|^2\right) + \alpha L^2$$

where L is the subgradient bound in Assumption 3, and $L(x, y)$ is the loss function defined in Equation (21), which shows that the loss function value of averaged iterates $L(\bar{x}_t, \bar{y}_t)$ converges to the saddle-point value $L(x^*, y^*)$ within error level αL^2 with rate $1/t$. It is noteworthy to mention that the error level can be further reduced by using a diminishing stepsize.

4.11 Empirical Results

We now demonstrate the effectiveness of the RO-TD algorithm against other algorithms across a number of benchmark domains. LARS-TD [37], which is a popular second-order sparse learning algorithm, is used as the baseline algorithm for feature selection and TDC is used as the off-policy convergent RL baseline algorithm, respectively.

4.12 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 w.r.t any transition is zero. Because of this, the Star MDP is unsuitable for LSTD-based algorithms, including LARS-TD. The random-walk problem [80] is a standard

Markov chain with 5 states and two absorbing state at two ends. Three sets of different bases Φ are used in [82], which are tabular features, inverted features and dependent features respectively. An identical experiment setting to [82] 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 subfigure of Figure 6, where TDC and RO-TD have almost identical MSPBE over iterations. The middle subfigure shows the value of $\langle y_t, Ax_t - b\rangle$ and $\|Ax_t - b\|_2$, wherein $\|Ax_t - b\|_2$ is always greater than the value of $\langle y_t, Ax_t - b\rangle$. 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 subfigure of Figure 6 shows that RO-TD is able to reduce MSPBE over successive iterations w.r.t three different basis functions.

4.13 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 MDP [80] 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 [37], 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 [37]. As the result shows in Table 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 focus of this paper 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 [77] which incorporates mirror descent and extragradient [40], as discussed below.

Algorithm	LARS-TD	RO-TD	l_2 LSTD	TDC	TD
Success$(20/20)$	100%	100%	0%	0%	0%
Steps	142.25 ± 9.74	147.40 ± 13.31	-	-	-

Table 1: Comparison of TD, LARS-TD, RO-TD, l_2 LSTD, TDC and TD

4.14 High-dimensional Under-actuated Systems

The triple-link inverted pendulum [75] 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 $\{0, 5\text{Newton}, -5\text{Newton}\}$. The goal is to learn a policy that can balance the arms for N_x steps within 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 initial state. We test the performance of RO-GQ(λ), GQ(λ) and LARS-TD. Two experiments are conducted with $N_x = 10,000$ and $100,000$, respectively. Fourier basis [39] with order 2 is used, i.e., 6561 basis. Table 2 shows the results of this experiment, where RO-GQ(λ) 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

Experiment\Method	RO-GQ(λ)	GQ(λ)	LARS-TD
Experiment 1	6.9 ± 4.82	11.3 ± 9.58	-
Experiment 2	14.7 ± 10.70	27.2 ± 6.52	-

Table 2: Comparison of RO-GQ(λ), GQ(λ), and LARS-TD on Triple-Link Inverted Pendulum Task showing minimum number of learning episodes.

of samples collected via random walk. We also find that tuning the sparsity parameter ρ_2 generates an interpolation between GQ(λ) and TD learning, where a large ρ_2 helps eliminates the correction term of TDC update and make the update direction more similar to the TD update. To sum up, RO-TD and RO-GQ(λ) tends to outperform GQ(λ) in all aspects, and is able to outperform LARS-TD 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 $O(Ndp^3)$, where that for RO-TD is $O(Nd)$. If p is linear or sublinear w.r.t d, RO-TD has a significant advantage over LARS-TD. However, compared with LARS-TD, RO-TD requires fine tuning the parameters of α_t, ρ_1, ρ_2 and is usually not as sample efficient as LARS-TD.

4.15 Conclusion

This section 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 is robust to noisy features. There are many interesting future directions for this research. One direction for future work is to extend the subgradient saddle-point solver to a more generalized mirror descent framework. Mirror descent is a generalization of subgradient descent with non-Euclidean distance [6], and has many advantages over gradient descent in high-dimensional spaces In [77], two algorithms to solve the bilinear saddle-point formulation are proposed based on mirror descent and extragradient [40], such as the Mirror-Prox algorithm. [77] also points out that the Mirror-Prox algorithm may be further optimized via randomization. To scale to larger MDPs, it is possible to design SMDP-based mirror-descent methods as well.

5 Sparse Nonlinear Value Function Approximation

Finally, we describe a new approach to representation discovery in reinforcement learning (RL) using basis adaptation. We introduce a general framework for basis adaptation as *nonlinear separable least-squares value function approximation* based on finding Fréchet gradients of an error function using variable projection functionals. We then present a scalable proximal gradient-based approach for basis adaptation using the recently proposed *mirror-descent* framework for RL. Unlike traditional temporal-difference (TD) methods for RL, mirror descent based RL methods undertake proximal gradient updates of weights in a *dual space*, which is linked together with the primal space using a Legendre transform involving the gradient of a strongly convex function. Mirror descent RL can be viewed as a proximal TD algorithm using Bregman divergence as the distance generating function. We present a new class of regularized proximal-gradient based TD methods, which combine feature selection through sparse L_1 regularization and basis adaptation. Experimental results are provided to illustrate and validate the approach.

5.1 Introduction

There has been rapidly growing interest in reinforcement learning in *representation discovery* [45]. *Basis construction* algorithms [46] combine the learning of features, or basis functions, and control. *Basis adaptation* [7, 53] enables tuning a given *parametric basis*, such as the Fourier basis [39], radial basis functions (RBF) [53], polynomial bases [41], etc. to the geometry of a particular Markov decision process (MDP). *Basis selection* methods combine sparse feature selection through L_1 regularization with traditional *least-squares* type RL methods [36, 27], linear complementarity methods [27], approximate linear programming [65], or convex-concave optimization methods for sparse off-policy TD-learning [42].

In this paper, we present a new framework for basis adaptation as nonlinear separable least-squares approximation of value functions using *variable projection functionals*. This framework is adapted from a well-known classical method for nonlinear regression [21], when the model parameters can be decomposed into a *linear set*, which is fit by classical least-squares, and a *nonlinear set*, which is fit by a Gauss-Newton method based on computing the gradient of an error function based on variable projection functionals.

Mirror descent is a highly scalable online convex optimization framework. Online convex optimization [88] explores the use of first-order gradient methods for solving convex optimization problems. Mirror de-

scent can be viewed as a first-order *proximal gradient* based method [3] using a distance generating function that is a Bregman divergence [13]. We combine basis adaptation with mirror-descent RL [47], a recently developed first-order approach to sparse RL. The proposed approach is also validated with some experiments showing improved performance compared to previous work.

5.2 Nonlinear Value Function Approximation

A key contribution of this paper is a general framework for nonlinear value function approximation based on *nonlinear separable least-squares*. This framework helps clarify previously proposed algorithms [7, 53], which amount to particular instances of this general framework. Concretely, basis adaptation can be understood as modifying a parameterized basis, $\Phi(\alpha)$ where α denotes the *nonlinear* parameters, and w denotes the *linear* parameters, such that $\hat{V} \approx \Phi(\alpha)w$. For example, for radial basis function (RBF) bases, α would correspond to the mean (center), μ_i, and covariance (width) parameters, Σ_i, of a fixed set of bases, where these would be tuned based on optimizing some particular error, such as the Bellman error [53]. Such an approach can broadly be characterized using the framework of separable nonlinear least squares framework for regression [21], which provides a framework for unifying past work on basis adaptation. The general *nonlinear regression* problem is to estimate a vector of parameters θ from a set of training examples $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ to minimize (for example) the squared error:

$$J(\theta) = \sum_{i=1}^{n} \|y_i - f(x_i; \theta)\|^2.$$

In the *nonlinear separable least squares setting*, the function $f(x_i; \theta) = f(x_i; \alpha, w) = [\Phi(\alpha)w]_i$, where α are the *nonlinear variables* and w denote the *linear variables*. The squared error can now be written as:

$$J(\alpha, w) \quad = \quad \|y - \Phi(\alpha)w\|^2 = \| \left(\mathbf{I} - \mathbf{\Phi}(\alpha)\mathbf{\Phi}^\dagger(\alpha)\right) \mathbf{y}\|^{\mathbf{2}}$$

Golub and Pereyra define the last (boldfaced) term above as the *variable projection functional* (VP). To explain the derivation of the second step from the first, note that for a specified value of the nonlinear parameters α, for the best linear fit, the linear parameter values can computed as $\hat{w} = \Phi^\dagger(\alpha)y$ (here, $\dagger$ denotes the Moore-Penrose pseudo-inverse). Geometrically, note that

$$\Pi^{\perp}_{\Phi(\alpha)} = (I - \Phi(\alpha)\Phi^\dagger(\alpha))$$

is the projector onto the complement of the column space of the parameterized basis matrix $\Phi(\alpha)$. Using this notation, the nonlinear separable least squares problem can be stated formally as:

$$\alpha_{\textsf{opt}} = \arg\min_{\alpha} \|\Pi^{\perp}_{\Phi(\alpha)} y\|^2 \tag{37}$$

If this problem is solved to find the optimal $\alpha_{\textsf{opt}}$, the optimal linear parameters $w_{\textsf{opt}} = \Phi^\dagger(\alpha_{\textsf{opt}})y$. To argue that this strategy is correct, Golub and Pereyra prove the following result:

Theorem 1. *[21] Assume that $\Phi(\alpha)$ has constant rank r in every open set $\Omega \subset \mathbb{R}^k$, where $\alpha \in \Omega$. If (α^*, w^*) is the global minimizer of $J(\alpha, w)$, then $\alpha_{\textsf{opt}} = \alpha^*$ and $w_{\textsf{opt}} = w^*$.*

To solve the optimization problem given by Equation 37, Golub and Pereyra derive a modified Gauss-Newton algorithm. The key step in their algorithm involves computation of the Jacobian $\nabla \|\Pi^{\perp}_{\Phi(\alpha)} y\|^2$, which involves finding the gradient of the pseudo inverse as follows:

$$\frac{1}{2}\nabla \|\Pi^{\perp}_{\Phi(\alpha)} y\|^2 = -\langle y, \Pi^{\perp}_{\Phi(\alpha)} D\Phi(\alpha)\Phi^\dagger(\alpha)y\rangle$$

where $D\Phi(\alpha)$ is the *Fréchet* (or tensor) derivative of the parameterized basis matrix $\Phi(\alpha)$. If there are m basis functions ϕ_i (columns of $\Phi(\alpha)$), n training points, and k nonlinear parameters α, then $D\Phi(\alpha)$ is a *tensor* of size $n \times m \times k$.

5.3 Bellman Residual VP Functional

The *Nonlinear Bellman Residual minimization* problem is to find a set of nonlinear weights $\alpha_{BR} \in \mathbb{R}^k$ and linear weights w_{BR} such that[5]

$$(\alpha_{BR}, w_{BR}) = \text{argmin}_{w,\alpha} \|T^\pi(\hat{V}) - \hat{V}\|^2_{\rho^\pi} \tag{38}$$

where $\hat{V} = \Phi(\alpha)w \approx V^\pi$, and the norm $\|x\|^2_{\rho^\pi} \equiv \langle x, D_{\rho^\pi} x\rangle$ where D_{ρ^π} is a diagonal matrix whose entries reflect the stationary distribution of the Markov chain corresponding to policy π. To lighten the notation below, we will assume $D_{\rho^\pi} = I$, although our entire derivation carries through without change in the more general case.

Defining $\Psi^\pi_\alpha = ((I - \gamma P^\pi)\Phi(\alpha))$, we can now eliminate the linear variables w_{NBR} and write the nonlinear Bellman residual variable projection functional as follows: $\alpha_{NBR} =$

$$\begin{aligned} &= \arg\min_\alpha \|R^\pi + \gamma P^\pi \hat{V} - \hat{V}\|^2 \\ &= \arg\min_\alpha \|R^\pi + (\gamma P^\pi \Phi(\alpha) - \Phi(\alpha))(\Psi^\pi_\alpha)^\dagger R^\pi\|^2 \\ &= \arg\min_\alpha \| \left[I - (I - \gamma P^\pi)\Phi(\alpha)(\Psi^\pi_\alpha)^\dagger\right] R^\pi\|^2 \end{aligned}$$

We can rewrite the final result above as $\alpha_{NBR} =$

$$\arg\min_\alpha \|\Pi_{\Psi^\pi_\alpha} R^\pi\|^2 = \arg\min_\alpha \|(I - \Psi^\pi_\alpha(\Psi^\pi_\alpha)^\dagger)R^\pi\|^2. \tag{39}$$

Given α_{NBR}, the linear weights w_{NBR} can be written as:

$$w_{NBR} = \Psi^\perp_{\Phi(\alpha_{NBR})} R^\pi = ((I - \gamma P^\pi)\Phi(\alpha_{NBR}))^\dagger R^\pi$$

Comparing with Equation 37, the optimization problem for nonlinear Bellman residual minimization once again involves minimization of a projector onto the complement of a matrix column space, in this case Ψ^π_α. We can state a new result extending [21] to the RL setting:

Theorem 2. *Assume that Ψ^π_α has constant rank r in every open set $\Omega \subset \mathbb{R}^k$, where $\alpha \in \Omega$. If (α_{BR}, w_{BR}) is the global minimizer of Equation 38, then $\alpha_{NBR} = \alpha_{BR}$ and $w_{NBR} = w_{BR}$.*

5.4 Nonlinear LSPI

We briefly describe a nonlinear variant of least-squares policy iteration (LSPI), where on each pass through the training data $\mathcal{D} = \{s_t, a_t, r_t, s_{t+1}\}$, a nonlinear optimization problem involving the best selection of the nonlinear parameters α is computed by solving (a sampled version of) Equation 39. We used `lsqnonlin`, a nonlinear least-squares solver within MATLAB, to solve a sampled variant of Equation 39, where $\Phi(\alpha_t)$ and $P^\pi\Phi(\alpha_t)$ were approximated by their sampled counterparts $\hat{\Phi}(\alpha_t)$ and $\hat{\Phi}'(\alpha_t)$. Once $\hat{\alpha}_{NBR}$ was found by the nonlinear phase, we used LSPI with these settings to find the linear weights $\hat{w}_{NBR}$. Figure 7 illustrates the application of this approach to a simple 25 state chain domain [41]. The reward is at the center, so the optimal policy is to go right in the first half of the chain and go left otherwise. The Q function is approximated by a set of radial basis functions (RBFs), whose initial centers and widths are specified as shown. The figure also shows the final modified centers and widths of the RBFs using the Bellman residual variable projection functional minimization method.

5.5 Mirror Descent RL

While the above framework of nonlinear separable value function approximation using variable projection functionals is theoretically elegant, it can be computationally expensive to compute Fréchet gradients of the variable projection functional in Equation 39 in large MDPs. We describe a novel mirror-descent gradient-based approach to basis adaptation that shows how to combine nonlinear value function approximation and sparsity.

[5] Similar VP functionals can be defined for other loss functions, e.g. for the fixed point or TD case.

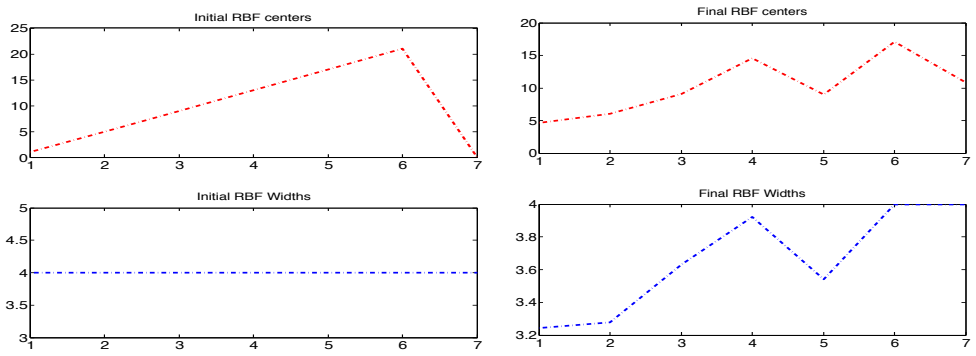


Figure 7: This figure shows the results of nonlinear basis adaptation by solving the Bellman residual variable projection functional represented by Equation 39 to find the optimal setting for the RBF centers and widths in a simple 25 state chain domain. Top two figures: initial centers and widths of RBF bases. Bottom two figures: Final centers and widths after convergence of nonlinear LSPI to the optimal policy.

5.6 Proximal Mappings and Mirror Descent

Mirror descent can be viewed as a first-order proximal gradient method [3, 88] for solving high-dimensional convex optimization problems. The simplest online convex algorithm is based on the classic gradient descent procedure for minimizing a convex function f, given as:

$$w_0 \in X, w_t = \Pi_X \left(w_{t-1} - \beta_t \nabla f(w_{t-1}) \right) : t \geq 1 \tag{40}$$

where $\Pi_X(x) = \text{argmin}_{y \in X} \|x - y\|^2$ is the projector onto set X, and β_t is a stepsize. If f is not differentiable, then the subgradient ∂f can be substituted instead, resulting in the well-known projected subgradient method, a workhorse of nonlinear programming. The proximal mapping associated with a convex function h is defined as:

$$\text{prox}_h(x) = \text{argmin}_u \left(h(u) + \|u - x\|_2^2 \right)$$

If $h(x) = 0$, then $\text{prox}_h(x) = x$, the identity function. If $h(x) = I_C(x)$, the indicator function for a convex set C, then $\text{prox}_{I_C}(x) = \Pi_C(x)$, the projector onto set C. For learning sparse representations, the case when $h(w) = \nu \|w\|_1$, for some scalar weighting factor ν, is particularly important. In this case, the entry-wise proximal operator is:

$$\text{prox}_h(w)_i = \begin{cases} w_i - \nu, & \text{if } w_i > \nu \\ 0, & \text{if } |w_i| \leq \nu \\ w_i + \nu & \text{otherwise} \end{cases} \tag{41}$$

An interesting observation follows from noting that the projected subgradient method (Equation 40) can be written equivalently using the proximal mapping as:

$$w_{t+1} = \text{argmin}_{w \in X} \left(\langle w, \partial f(w_t) \rangle + \frac{1}{2\beta_t} \|w - w_t\|_2^2 \right) \tag{42}$$

An intuitive way to understand this equation is to view the first term as requiring the next iterate w_{t+1} to move in the direction of the (sub) gradient of f at w_t, whereas the second term requires that the next iterate w_{t+1} not move too far away from the current iterate w_t. Note that the (sub)gradient descent is a special case of Equation (42) with Euclidean distance metric. With this introduction, we can now introduce the main concept of *mirror descent*. We follow the treatment in [3] in presenting the mirror descent algorithm as a nonlinear proximal method based on a distance generating function that is a Bregman divergence [13].

Definition 1: A distance generating function $\psi(x)$ is defined as a continuously differentiable strongly convex function (with modulus σ) which satisfies:

$$\langle x' - x, \nabla \psi(x') - \nabla \psi(x) \rangle \geq \sigma \|x' - x\|^2 \tag{43}$$

Given such a function ψ, the Bregman divergence associated with it is defined as:

$$D_\psi(x, y) = \psi(x) - \psi(y) - \langle \nabla\psi(y), x - y \rangle \tag{44}$$

Intuitively, the Bregman divergence measures the difference between the value of a strongly convex function $\psi(x)$ and the estimate derived from the first-order Taylor series expansion at $\psi(y)$. Many widely used distance measures turn out to be special cases of Bregman divergences, such as Euclidean distance (where $\psi(x) = \frac{1}{2}\|x\|_2^2$) and Kullback Liebler divergence (where $\psi(x) = \sum_i x_i \log_2 x_i$, the negative entropy function). In general, Bregman divergences are non-symmetric, but projections onto a convex set with respect to a Bregman divergence are well-defined.

The general mirror descent procedure can be written as:

$$w_{t+1} = \text{argmin}_{w \in X} \left(\langle w, \partial f(w_t) \rangle + \frac{1}{\alpha_t} D_\psi(w, w_t) \right) \tag{45}$$

Notice that the squared distance term in Equation 42 has been generalized to a Bregman divergence. The solution to this optimization problem can be stated succinctly as the following generalized gradient descent algorithm, which forms the core procedure in mirror descent:

$$w_{t+1} = \nabla\psi^* \left(\nabla\psi(w_t) - \alpha_t \partial f(w_t) \right) \tag{46}$$

Here, ψ^* is the Legendre transform of the strongly convex function ψ, which is defined as

$$\psi^*(y) = \sup_{x \in X} \left(\langle x, y \rangle - \psi(x) \right)$$

It can be shown that $\nabla\psi^* = (\nabla\psi)^{-1}$ [3]. Mirror descent is a powerful first-order optimization method that been shown to be "universal" in that if a problem is online learnable, it leads to a low-regret solution using mirror descent [78]. It is shown in [5] that the mirror descent procedure specified in Equation 46 with the Bregman divergence defined by the *p-norm* function, defined below, can outperform the projected subgradient method by a factor $\frac{n}{\log n}$ where n is the dimensionality of the space. For high-dimensional spaces, this ratio can be quite large.

5.7 Sparse Learning with Mirror Descent TD

Recently, a new framework for sparse regularized RL was proposed based on mirror descent [47]. Unlike regular TD, the weights are updated using the TD error in the dual space by mapping the primal weights w using a gradient of a strongly convex function ψ. Subsequently, the updated dual weights are converted back into the primal space using the gradient of the Legendre transform of ψ, namely $\nabla\psi^*$. To achieve sparsity, the dual weights θ are truncated according to Equation 41 to satisfy the l_1 penalty on the weights. Here, ν is the sparsity parameter. An analogous approach for l_1 penalized classification and regression was suggested in [74].

One choice for Bregman divergence is the **p-norm** function, where $\psi(w) = \frac{1}{2}\|w\|_q^2$, and its conjugate Legendre transform $\psi^*(w) = \frac{1}{2}\|w\|_p^2$. Here, $\|w\|_q = \left(\sum_j |w_j|^q \right)^{\frac{1}{q}}$, and p and q are conjugate numbers such that $\frac{1}{p} + \frac{1}{q} = 1$. This $\psi(w)$ leads to the p-norm link function $\theta = f(w)$ where $f : \mathbb{R}^d \to \mathbb{R}^d$:

$$f_j(w) = \frac{\text{sign}(w_j)|w_j|^{q-1}}{\|w\|_q^{q-2}}, \quad f_j^{-1}(\theta) = \frac{\text{sign}(\theta_j)|\theta_j|^{p-1}}{\|\theta\|_p^{p-2}} \tag{47}$$

Many other choices for Bregman divergence are possible.

5.8 Basis Adaptation with Mirror Descent RL

Our method is based on the two-time scale stochastic approximation framework, whereby the linear parameters w are adapted at a faster time-scale than the nonlinear parameters α. Algorithm 1 below describes

the nonlinear adaptive basis mirror-descent variant of Watkins $Q(\lambda)$ algorithm.[6] We indicate the dynamically varying nature of the bases as $\phi_t(s_t, a_t)$ where the subscript denotes the particular value of the nonlinear parameters α_t at time t. In this section, we denote β_t as the learning rate for the faster time-scale update procedure for updating the linear weights w_t and ξ_t as the learning rate for the slower time-scale parameter for updating the nonlinear basis parameters α_t. The key idea underlying Algorithm 1 is to adapt the nonlinear parameters using the gradient of the TD error. As the gradient of the TD error is not easily computed due to the `max` computation in $Q(\lambda)$, what is done here is to approximate the maximum using a smoothed differentiable approximation:

$$\max_{i=1,\ldots,n} x_i \approx f_\sigma(\{x_i\}_{i=1,\ldots,n}) = \log(\sum_{i=1}^{n} e^{\sigma x_i})$$

We update the nonlinear basis parameters α_t on a slower time scale using the gradient of the smoothed TD error. The mirror-descent version of Watkins $Q(\lambda)$ with basis adaptation is given below.

Algorithm 3 Mirror Descent Q(λ) with Basis Adaptation

Given: Parametric basis $\Phi(\alpha)$

Initialize linear parameters w_0 and nonlinear parameters α_0.

repeat

Do action $a_t = \text{argmax}_{a^*} \langle \phi_t(s_t, a^*), w_t \rangle$ with high probability, else do a random action. Observe next state s_{t+1} and reward r_t.

Compute the actual TD error

$$\delta_t = r_t + \gamma \max_a \langle \phi_t(s_{t+1}, a), w_t \rangle - \langle \phi_t(s_t, a_t), w_t \rangle$$

Compute the smoothed TD error

$$\omega_t = r_t + \gamma f_\sigma(\{\langle \phi_t(s_{t+1}, a'), w_t \rangle\}_{a'}) - \langle \phi_t(s_t, a_t), w_t \rangle$$

Compute the gradient of the smoothed TD error w.r.t.. α

$$K_\sigma(s_t, a_t, s_{t+1}) = \gamma \sum_{a'} \frac{\partial f_\sigma(\{\langle \phi_t(s_{t+1}, a'), w_t \rangle\}_{a'})}{\partial(\langle \phi_t(s_{t+1}, a'), w_t \rangle)}$$

$$\times \nabla_\alpha(\langle \phi_t(s_{t+1}, a'), w_t \rangle) - \nabla_\alpha(\langle \phi_t(s_{t+1}, a_t), w_t \rangle)$$

Update the eligibility trace $e_t \leftarrow e_t + \lambda\gamma\phi_t(s_t, a_t)$

Update the dual weights $\theta_t = \nabla\psi_t(w_t) + \beta_t\delta_t e_t$ (e.g., $\psi(w) = \frac{1}{2}\|w\|_q^2$ is the p-norm link function).

Truncate weights:

$$\forall j, \quad \theta_j^{t+1} = \text{sign}(\tilde{\theta}_j^{t+1}) \max(0, |\tilde{\theta}_j^{t+1}| - \beta_t \nu)$$

Update linear weight parameters: $w_{t+1} = \nabla\psi_t^*(\theta_{t+1})$ (e.g., $\psi^*(\theta) = \frac{1}{2}\|\theta\|_p^2$ and p and q are dual norms such that $\frac{1}{p} + \frac{1}{q} = 1$).

Update nonlinear basis parameters:

$$\alpha_{t+1} \leftarrow \alpha_t + \xi_t \omega_t K_\sigma$$

Set $t \leftarrow t + 1$.

until done.

Return $\hat{Q}(s, a) = (\Phi(\alpha_t)w_t)_{s,a}$ as the approximate l_1 penalized sparse optimal action value function.

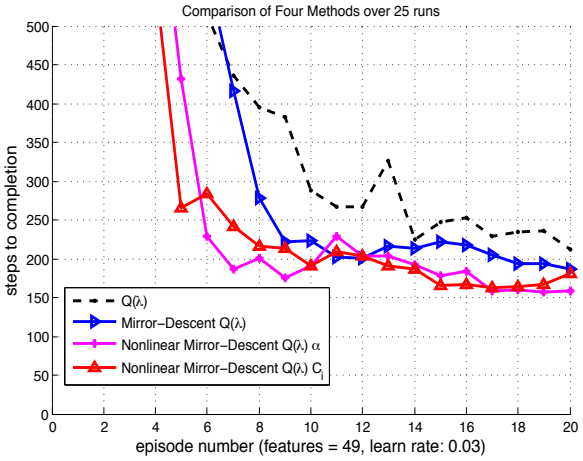


Figure 8: Comparing the convergence of two mirror-descent $Q(\lambda)$ methods over 25 runs with 49 (α and c_i parameter tuned) nonlinear Fourier bases in the mountain car task with that of $Q(\lambda)$, and mirror-descent $Q(\lambda)$, with 49 fixed Fourier bases.

5.9 Experimental Results

In this section, we provide some illustrative experiments evaluating the performance of the mirror-descent $Q(\lambda)$ method with a nonlinear generalization of the fixed Fourier basis [39]. The nonlinear Fourier basis that we used is as follows:

$$\phi_i(\mathbf{x}) = \cos(\pi\alpha_i\langle c_i, \mathbf{x}\rangle)$$

The nonlinear parameters here include both α_i and c_i. Note that in the previous work on Fourier bases [39], the scaling factor α_i was fixed to unity, whereas in our case, it is allowed to vary (as shown in Figure 10). Also, earlier work set $c^i = [c_1, \ldots, c_d], c_j \in [0, \ldots, n], 1 \leq j \leq d$. For example, in the mountain car domain, $d = 2$, and setting $n = 3$ would result in 16 possible features (since $n = 0, 1, 2, 3$). The nonlinear parameter α_i can be seen here as a weighting of how important each c_i is. In our initial experiments, we vary both α and c_i separately to compare their effects.

Figure 8 compares the convergence and stability of $Q(\lambda)$ and mirror-descent $Q(\lambda)$ using 49 fixed Fourier bases with mirror-descent $Q(\lambda)$ using the same number of adaptive (α and C_i parameter tuned) nonlinear Fourier bases. In this case, all methods converge, but the two nonlinear mirror-descent methods (bottom curves) converge much more quickly. The mirror-descent methods have converged by 7 episodes to an average solution length at least thrice as small as the regular $Q(\lambda)$ method, which takes much longer to reliably achieve the same solution quality. For the experimental result shown in Figure 8, where we show convergence in the mountain car domain with 49 Fourier bases, we set $\beta_t = 0.03$, $\xi_t = 5 \times 10^{-7}$, and $\nu = 0.001$. The **p-norm** Bregman divergence was used, where $p = 2\log_{10}(\text{num\_features x num\_actions})$, and $q = \frac{p}{p-1}$. The learning rate for $Q(\lambda)$ had to be set fairly low at 0.001 when the number of features exceeded 16 as higher rates caused instability.

Figure 9 shows the variance across 25 runs of the four methods compared in Figure 8. Note the improved variance of the two mirror-descent methods that use basis adaptation, showing they not only result in improved performance, but also lower variance. Finally, Figure 10 shows the ranges of the α parameters as tuned by the basis adaptation algorithm for 49 Fourier bases over 25 runs.

5.10 Conclusions and Future Work

This section introduced a broad framework for basis adaptation as nonlinear separable least-squares value approximation, and described a specific gradient-based method for basis adaptation using mirror descent

[6]The extension to related methods like $SARSA(\lambda)$ [80] is straightforward. Some details abbreviated for clarity.

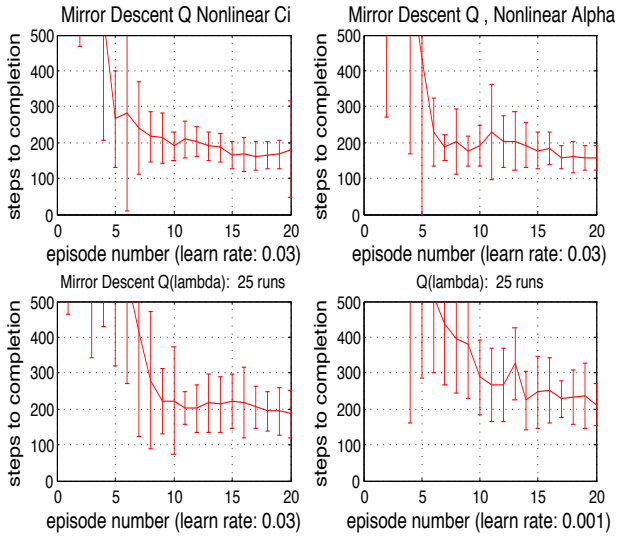


Figure 9: Variance of the four methods compared in Figure 8 across 25 runs. Note the improved variance of the two methods shown on top that combine mirror-descent updating and basis adaptation.

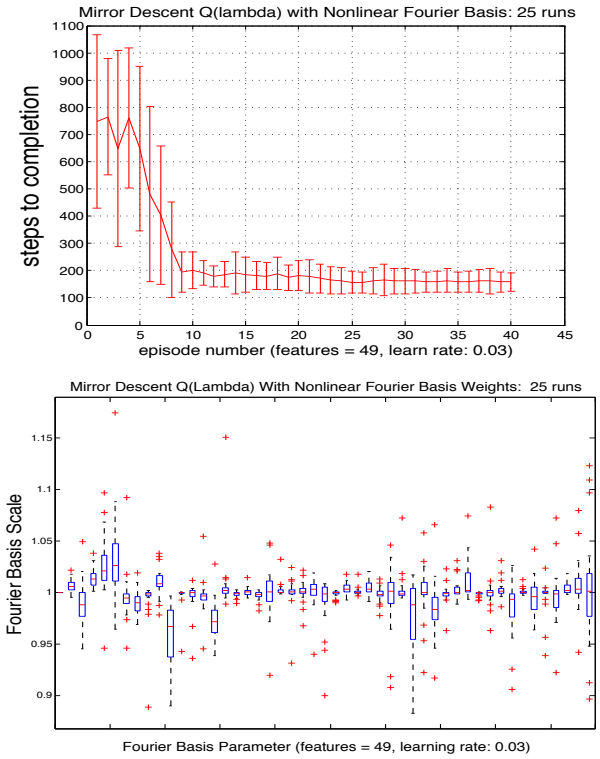


Figure 10: (a) Extended run of the mirror-descent $Q(\lambda)$ method with basis adaptation of the α parameters in the mountain car domain. (b) Boxplot of the α parameters for 49 Fourier bases for the run shown on top.

optimization. Currently, the learning rate for nonlinear parameters needs to be set quite low for stable

convergence. More rapid tuning of the nonlinear parameters could be achieved by including an additional mirror descent step for modifying them.

6 Summary

This research investigated the automated design of problem-specific representations for approximately solving Markov decision processes (MDPs), a widely used model of sequential decision making. The research carried out nvestigated a variety of approaches to automatic basis construction, including reward-sensitive and reward-invariant methods, diagonalization and dilation methods, as well as orthogonal and over-complete representations. A unifying perspective on the various basis construction methods emerges from showing they result from different power series expansions of value functions, including the Neumann series expansion, the Laurent series expansion, and the Schultz expansion.

The research also develops new computational algorithms for learning to solve Markov decision processes with an overcomplete representation. The main idea is to use a L_1 regularized penalty function that promotes finding sparse representations of value functions. A key idea investigated in this research is the use of *mirror-descent* optimization methods to find sparse representations of value functions. A new regularized off-policy bradient temporal-difference (TD) method is proposed, which combines off-policy convergence, robustness to irrelevant features, and scalability to large problems. Experimental results show promising performance.

References

[1] R. Agaev and P. Cheboratev. On the spectra of nonsymmetric Laplacian matrices. *Linear Algebra and its Applications*, 399:157–168, 2005.

[2] A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Systems Journal*, 13:41–77, 2003.

[3] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, Jan 2003.

[4] M Belkin and P Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 6(15):1373–1396, June 2003.

[5] A. Ben-Tal, T. Margalit, and A. Nemirovski. The ordered subsets mirror descent optimization method with applications to tomography. *SIAM Journal of Optimization*, Jan 2001.

[6] A. Ben-Tal and A. Nemirovski. Non-Euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102(3):407–456, 2005.

[7] D. Bertsekas and H. Yu. Basis function adaptation methods for cost approximation in Markov Decision Processes. In *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2009.

[8] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996.

[9] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2000.

[10] Gregory Beylkin, Ronald R Coifman, and Vladimir Rokhlin. Fast wavelet transforms and numerical algorithms. *Comm Pure Applied math*, 44:141–183, 1991.

[11] C. Bishop. *Machine Learning and Pattern Recognition*. Springer, 2007.

[12] S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

[13] L. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217, 1967.

[14] S. Campbell and C. Meyer. *Generalized Inverses of Linear Transformations*. Pitman, 1979.

[15] L. Chen, E. Krishnamurthy, and I. Macleod. Generalized matrix inversion and rank computation by repeated squaring. *Parallel Computing*, 20:297–311, 1994.

[16] Ronald Parr Christopher Painter-Wakefield. Greedy algorithms for sparse reinforcement learning. In *International Conference on Machine Learning*, 2012.

[17] F Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, April 2005.

[18] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, July 2006.

[19] Daniela Pucci de Farias and Benjamin Van Roy. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*, 51(6):850–865, 2003.

[20] Mohammad Ghavamzadeh, Alessandro Lazaric, Remi Munos, and Matthew Hoffman. Finite-Sample Analysis of Lasso-TD . In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, 2011.

[21] G. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal of Numerical Analysis*, 10:413–32, 1973.

[22] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of AI Research*, 19:399–468, 2003.

[23] Jesse Hoey, Robert St-aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 279–288. Morgan Kaufmann, 1999.

[24] R. Howard. *Dynamic Programming and Markov Decision Processes*. MIT Press, 1960.

[25] R. Howard. *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes (volume 2)*. Wiley, 1971.

[26] J. Johns and S. Mahadevan. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 385–392. ACM Press, 2007.

[27] J. Johns, C. Painter-Wakefield, and R. Parr. Linear complementarity for regularized policy evaluation and improvement. *In Proceedings of Advances in Neural Information Processing Systems*, 23, 2010.

[28] J. Johns, M. Petrik, and S. Mahadevan. Hybrid least-squares algorithms for approximate policy evaluation. *Machine Learning*, 76(2-3):243–256, 2009.

[29] Jeff Johns and Sridhar Mahadevan. Sparse approximate policy evaluation using graph-based basis functions. Technical Report UM-CS-2009-41, Department of Computer Science, University of Massachusetts Amherst, 2009.

[30] Jeff Johns, Christopher Painter-Wakefield, and Ronald Parr. Linear complementarity for regularized policy evaluation and improvement. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2010.

[31] T. Jolliffe. *Principal Components Analysis*. Springer-Verlag, 1986.

[32] A. Juditsky, A.S. Nemirovskii, and C. Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Arxiv preprint arXiv:0809.0815*, 2008.

[33] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[34] P. Keller, S. Mannor, and D Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 22^{nd} International Conference on Machine Learning (ICML)*, pages 449–456. MIT Press, 2006.

[35] D. Koller and N. Friedman. *Graphical Models*. MIT Press, 2009.

[36] J. Zico Kolter and Andrew Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 521–528, New York, NY, USA, 2009. ACM.

[37] J. Zico Kolter and Andrew Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of 27 th International Conference on Machine Learning*, 2009.

[38] J. Zico Z. Kolter. The Fixed Points of Off-Policy TD. In *Advances in Neural Information Processing Systems 24*, pages 2169–2177, 2011.

[39] G. Konidaris, S. Osentoski, and PS Thomas. Value function approximation in reinforcement learning using the fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 2011.

[40] G. M. Korpelevich. The extragradient method for finding saddle points and other problems. 1976.

[41] M. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[42] B. Liu, S. Mahadevan, and J. Liu. Regularized off-policy TD-learning. In *Neural Information Processing Systems Conference (NIPS)*, December 2012.

[43] H.R. Maei and R.S. Sutton. GQ (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence*, pages 91–96, 2010.

[44] Mauro Maggioni and Sridhar Mahadevan. Fast direct policy evaluation using multiscale analysis of markov diffusion processes. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 601–608, New York, NY, USA, 2006. ACM Press.

[45] S. Mahadevan. *Representation Discovery using Harmonic Analysis*. Morgan and Claypool Publishers, 2008.

[46] S. Mahadevan. Learning Representation and Control in Markov Decision Processes: New Frontiers. *Foundations and Trends in Machine Learning*, 1(4):403–565, 2009.

[47] S. Mahadevan and B. Liu. Reinforcement learning using mirror descent. In *UAI 2012: Proceedings of the conference on Uncertainty in AI*, 2012.

[48] S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *Proceedings of the Neural Information Processing Systems (NIPS)*. MIT Press, 2006.

[49] S. Mahadevan and M. Maggioni. Proto-Value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes. *Journal of Machine Learning Research*, 8:2169–2231, 2007.

[50] S. Mahadevan, M. Maggioni, K. Ferguson, and S. Osentoski. Learning representation and control in continuous markov decision processes. In *AAAI'06: proceedings of the 21st national conference on Artificial intelligence*, pages 1194–1199. AAAI Press, 2006.

[51] Sridhar Mahadevan. Proto-value functions: developmental reinforcement learning. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 553–560, New York, NY, USA, 2005. ACM.

[52] Alessandro Lazaric Mohammad Ghavamzadeh Matthieu Geist, Bruno Scherrer. A dantzig selector approach to temporal difference learning. In *International Conference on Machine Learning*, 2012.

[53] N. Menache, N. Shimkin, and S. Mannor. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134:215–238, 2005.

[54] R. Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1006–1011, 2005.

[55] A. Nedić and D. P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1-2):79–110, 2003.

[56] A. Nedić and A. Ozdaglar. Subgradient methods for saddle-point problems. *Journal of optimization theory and applications*, 142(1):205–228, 2009.

[57] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.

[58] A. Ng, H. Kim, M. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In *Proceedings of Neural Information Processing Systems*, 2004.

[59] S. Osentoski and S. Mahadevan. Learning State Action Basis Functions for Hierarchical Markov Decison Processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 705–712, 2007.

[60] S. Osentoski and S. Mahadevan. Basis function construction for hierarchical reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010.

[61] R. Parr, C. Painter-Wakefiled, L. Li, and M. Littman. Analyzing feature generation for value function approximation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 737–744, 2007.

[62] R. Parr, C. Painter-Wakefiled, L. Li, and M. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[63] J. Pearl. *Probabilistic Reasoning In Intelligent Systems*. Morgan-Kauffman, 1988.

[64] M. Petrik. An analysis of Laplacian methods for value function approximation in MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2574–2579, 2007.

[65] Marek Petrik, Gavin Taylor, Ronald Parr, and Shlomo Zilberstein. Feature selection using regularization in approximate linear programs for Markov decision processes. In *Proceedings of the International Conference on Machine learning (ICML)*, 2010.

[66] R. Platt, A. Fagg, and R Grupen. Manipulation gaits: Sequences of grasp control tasks. In *IEEE Conference on Robotics and Automation (ICRA)*, 2004.

[67] P. Poupart and C. Boutilier. Value directed compression of POMDPs. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2003.

[68] W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, 2007.

[69] M. L. Puterman. *Markov decision processes*. Wiley Interscience, New York, USA, 1994.

[70] K. Rohanimanesh and S. Mahadevan. Coarticulation: an approach for generating concurrent plans in markov decision processes. In *ICML*, pages 720–727, 2005.

[71] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2002.

[72] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM Press, 2003.

[73] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[74] S. Shalev-Shwartz and A. Tewari. Stochastic methods for l1 regularized loss minimization. *Journal of Machine Learning Research*, June 2011.

[75] Jennie Si and Yu-Tsung Wang. Online learning control by association and reinforcement. *IEEE Transactions on Neural Networks*, 12:264–276, 2001.

[76] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over the finite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

[77] S. Sra, S. Nowozin, and S.J. Wright. *Optimization for Machine Learning*. MIT Press, 2011.

[78] N. Srebro, K. Sridharan, and A. Tewari. On the universality of online mirror descent. *Arxiv preprint arXiv:1107.4080*, Jan 2011.

[79] Masashi Sugiyama, Hirotaka Hachiya, Christopher Towell, and Sethu Vijayakumar. Geodesic gaussian kernels for value function approximation. *Autonomous Robots*, 25(3):287–304, 2008.

[80] R. Sutton and A. G. Barto. *An Introduction to Reinforcement Learning*. MIT Press, 1998.

[81] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[82] R.S. Sutton, H.R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning*, pages 993–1000, 2009.

[83] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–278, 1992.

[84] G. Theocharous and S. Mahadevan. Compressing POMDPs using Locality Preserving Non-Negative Matrix Factorization. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, Atlanta, 2010. AAAI Press/MIT Press.

[85] Richard Sutton Thomas Degris, Martha White. Linear off-policy actor-critic. In *International Conference on Machine Learning*, 2012.

[86] Y. Wei. Successive matrix squaring algorithm for computing the Drazin inverse. *Applied Mathematics and Computation*, 108:67–75, 2000.

[87] H. Yu and D. Bertsekas. Basis function adaptation methods for cost approximation in MDPs. In *Proceedings of the IEEE International Symposium on Dynamic Programming and Reinforcement Learning*, 2009.

[88] M Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.