Sparse Reinforcement Learning via Convex Optimization

Zhiwei (Tony) Qin[1] TQIN@WALMARTLABS.COM
@WalmartLabs, 850 Cherry Ave, San Bruno, CA 94066

Weichang Li, Firdaus Janoos WEICHANG.LI, FIRDAUS.JANOOS@EXXONMOBIL.COM
ExxonMobil Corporate Strategic Research, 1545 Rt. 22 East, Annandale, NJ 08801

Abstract

We propose two new algorithms for the sparse reinforcement learning problem based on different formulations. The first algorithm is an off-line method based on the alternating direction method of multipliers for solving a constrained formulation that explicitly controls the projected Bellman residual. The second algorithm is an online stochastic approximation algorithm that employs the regularized dual averaging technique, using the Lagrangian formulation. The convergence of both algorithms are established. We demonstrate the performance of these algorithms through several classical examples.

1. Introduction

As an approximation technique for problems involving Markov Decision Processes (MDP), reinforcement learning has been applied to such diverse fields as robotics (Bentivegna et al., 2002), helicopter control (Ng et al., 2006), and operations research (Proper & Tadepalli, 2006), among many others. The problem of approximating the value function of an MDP is central in reinforcement learning, where the model information is usually unavailable, and even noisy samples of the true target function values are not accessible. When the state space is large or infinite-dimensional, such a task is often accomplished through the linear approximation architecture, under which the hypothesis space is spanned by a set of K basis functions or 'features' (Tsitsiklis & Van Roy, 1996). The least-squares temporal difference (LSTD) (Bradtke & Barto, 1996) learning and gradient-based TD learning (Sutton et al., 2009) have been effective ways of learning a linear representation of the value function by using realized trajectory samples. It has become increasingly clear that learning a sparse representation of the value function can prevent over-fitting and improve generalization performance, especially with the presence of a large number of features or a limited number of samples (e.g., (Kolter & Ng, 2009)).

In this paper we approach the sparse reinforcement learning problem with a new constrained formulation that explicitly controls the projected Bellman residual (PBR) and a popular Lagrangian formulation based on l_1-regularization. We propose tailored optimization algorithms for solving each of these two sparse reinforcement learning problems, including: 1) an efficient off-line off-policy learning algorithm using the alternating direction method of multipliers (ADMM) (Eckstein & Bertsekas, 1992) with a proximal gradient step, and 2) a novel online stochastic approximation algorithm that combines elements from the regularized dual-averaging method (Xiao, 2010) and the least mean squares technique. Convergence results are established for both algorithms. We demonstrate through experiment results that regularized PBR minimization might be preferable over regularized minimization of the fixed-point slack for value function approximation.

The rest of this paper is organized as follows. Section 2 reviews reinforcement learning with approximate value and policy iterations, and TD learning. The main contribution of this paper is presented in Sections 3 and 4. Section 3 introduces the constrained and the Lagrangian formulations for sparse value function approximation and their extensions to the Q-function. The two new off-policy learning algorithms and their convergence proof are presented in Section 4. We discuss related existing work in Section 4.4 and present experiment results against LSTD and two other sparse reinforcement learning algorithms in Section 5. Finally we conclude the paper in Section 6.

2. Background and Preliminaries

In a Markov Decision Process (MDP), the state-action pair is denoted by $(\mathbf{x}, u) \in \mathcal{X} \times \mathcal{U}(\mathbf{x})$, where $\mathcal{X}$ and $\mathcal{U}(\mathbf{x})$ are the

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

[1]The work was completed when the first author was a summer intern with ExxonMobil Corporate Strategic Research in 2012 working towards a Ph.D. degree at Columbia University, New York.

set of states and the set of compatible actions respectively. The set of state indices is $\mathcal{S}$. The immediate reward by applying u to $\mathbf{x}$ is $r(\mathbf{x}, u)$. We use the capital letters X and R to denote the random variable versions of $\mathbf{x}$ and r. Hence the random reward $R(t) := R(X(t), \pi(X(t)))$. π denotes a particular policy mapping from a state to a compatible action, i.e. $\pi(\mathbf{x}) \in \mathcal{U}(\mathbf{x})$. The value function, $V^\pi : \mathcal{X} \to \mathbb{R}$ is defined as $V^\pi(\mathbf{x}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(t) \mid X(0) = \mathbf{x}\right]$, with the discounting factor $\gamma \in [0, 1)$. V^π satisfies the Bellman's equation: for $\mathbf{x} \in \mathcal{X}$,

$$V^\pi(\mathbf{x}) = r(\mathbf{x}, \pi(\mathbf{x})) + \gamma \sum_{\mathbf{y} \in \mathcal{X}} \mathcal{P}(\mathbf{x}, \pi(\mathbf{x}), \mathbf{y}) V^\pi(\mathbf{y}), \quad (1)$$

where $\mathcal{P}(\mathbf{x}, u, \mathbf{y})$ is the transition probability from state $\mathbf{x}$ to state $\mathbf{y}$ by applying action u. The transition probabilities and the reward distribution are generally not known. The optimal value function $V^*(\mathbf{x})$ satisfies the Bellman's optimality equation:

$$V^*(\mathbf{x}) = \max_{u \in \mathcal{U}(\mathbf{x})} r(\mathbf{x}, u) + \gamma \sum_{\mathbf{y} \in \mathcal{X}} \mathcal{P}(\mathbf{x}, u, \mathbf{y}) V^*(\mathbf{y}) \quad (2)$$

Define the Bellman operator T^π by the right-hand-side of (1) so that $T^\pi(V^\pi) := V^\pi$, and similarly, the Bellman optimality operator T^* by the right-hand-side of (2) so that $T^*(V^*) := V^*$.

2.1. Value Function Approximation

For problems with a large state space, evaluating $V^{(t)}$ exactly can be infeasibly expensive. A natural approach is to approximate V by $\tilde{V}$ through some compact representation with a parameter $\boldsymbol{\beta} \in \mathbb{R}^K$. The most common approximation architecture is linear function approximation (Tsitsiklis & Van Roy, 1996), i.e.

$$\tilde{V}^{(t)}(\mathbf{x}) = \sum_{k=1}^{K} \boldsymbol{\beta}_k \phi_k(\mathbf{x}) = \Phi(\mathbf{x})^T \boldsymbol{\beta} \quad (3)$$

$\boldsymbol{\phi}(\mathbf{x})$ defines a K-dimensional feature vector for the state $\mathbf{x}$ and is part of the design architecture.

2.2. Approximate Policy Iteration

We consider the following state-action value function

$$Q^\pi(\mathbf{x}, u) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^{(t+1)} | X^{(0)} = \mathbf{x}, U^{(0)} = u\right],$$

which offers the flexibility of specifying the initial action u. In approximate Policy Iteration, the tasks of policy evaluation and policy improvement are separated. Given a particular policy π, we approximate the state-action value function $Q^\pi(\mathbf{x}, u)$ by using function approximation techniques. An improved policy π^+ is then built upon $Q^\pi(\mathbf{x}, u)$, and the iterations continue till convergence. Our approach for learning the optimal policy falls into this class of methods.

2.3. Temporal Difference (TD) Learning

With the transition probabilities and the reward distribution unknown *a priori*, the value function V^π with regard to the underlying policy π has to be learned through a set of n transition observations (or samples) $\{(\mathbf{x}_t, u_t, r_{t+1}, \mathbf{y}_{t+1})\}_{t=1}^n$. The most common approaches include TD Learning (Sutton, 1988) and the Least Squares TD (LSTD) (Bradtke & Barto, 1996). Defining the temporal difference associated with the transition from time t to $t+1$ by

$$\delta_{t+1} = r_{t+1} + \gamma \tilde{V}^{(t)}(\mathbf{y}_{t+1}) - \tilde{V}^{(t)}(\mathbf{x}_t), \quad (4)$$

TD Learning with linear value function approximation (3) takes the following form (Tsitsiklis & Van Roy, 1997):

$$\begin{cases} \delta_{t+1}(\boldsymbol{\beta}) = r_{t+1} + \gamma \boldsymbol{\phi}'^{\top}_{t+1} \boldsymbol{\beta} - \boldsymbol{\phi}_t^\top \boldsymbol{\beta} \\ \boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \alpha_t \delta_{t+1} \nabla_\beta \tilde{V}^{(t)}(\mathbf{x}_t). \end{cases} \quad (5)$$

where the short-hand notation $\boldsymbol{\phi}_t := \boldsymbol{\phi}(\mathbf{x}_t)$, $\boldsymbol{\phi}'_{t+1} := \boldsymbol{\phi}(\mathbf{y}_{t+1})$ denote the feature vectors.

The LSTD algorithm solves the linear system

$$\tilde{\mathbf{A}}^{(n)} \boldsymbol{\beta} = \tilde{\mathbf{b}}^{(n)} \quad (6)$$

where $\tilde{\mathbf{A}}^{(n)} := \frac{1}{n} \sum_{t=1}^n (\boldsymbol{\phi}_t (\boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}'_{t+1})^\top)$, $\tilde{\mathbf{b}}^{(n)} := \frac{1}{n} \sum_{t=1}^n (r_{t+1} \boldsymbol{\phi}_t)$. We will drop the superscripts for simpler notation. Loosely speaking, the overall idea here is to sample the unknown state transition probabilities from a sequence of state trajectories, and to construct the subspace within which the value function could be well approximated in linear form.

For compact notation, we replace the summation terms in (6) by the matrix forms:

$$\tilde{\mathbf{A}} := \frac{1}{n} (\tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{\Phi}} - \gamma \tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{\Phi}}') \quad (7)$$

$$\tilde{\mathbf{b}} := \frac{1}{n} (\tilde{\boldsymbol{\Phi}}^\top \tilde{R}). \quad (8)$$

where $\tilde{\boldsymbol{\Phi}} := \left(\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_n\right)^\top \in \mathbb{R}^{n \times K}$, $\tilde{\boldsymbol{\Phi}}' := \left(\boldsymbol{\phi}'_2, \cdots, \boldsymbol{\phi}'_{n+1}\right)^\top \in \mathbb{R}^{n \times K}$, $\tilde{R} := \left(r_2, \cdots, r_{n+1}\right)^\top \in \mathbb{R}^n$. In addition, we define $\tilde{\mathbf{G}} := n(\tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{\Phi}})^{-1}$. We note that $\tilde{\mathbf{A}}$ $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{G}}$ may be viewed as sample averaged approximation to the following ensemble avaerage terms: $\mathbf{A} := \mathbb{E}[\boldsymbol{\phi}(X)(\boldsymbol{\phi}(X) - \gamma \boldsymbol{\phi}(Y))^\top]$, $\mathbf{b} := \mathbb{E}[R\boldsymbol{\phi}(X)]$ and $\mathbf{G} := \mathbb{E}[\boldsymbol{\phi}(X)\boldsymbol{\phi}(X)^\top]^{-1}$.

3. Sparse Reinforcement Learning

The linear value function approximation architecture poses two conflicting requirements on the size of the feature space: First, a compact representation of the value function, desired for both computational and storage consideration, calls for a small number of features; On the other

hand, a better approximation to the value function requires a rich feature space which implies a relatively large number of features. However, as we saw above, the linear system for LSTD may not even be invertible with the number of features larger than the number of samples. This motivates feature selection, or more precisely, learning a sparse representation out of a large number of basis functions.

A common loss function, which both the GTD2/TDC (Sutton et al., 2009) and LSTD methods minimize, is based on the *projected Bellman residual* (PBR): $L_{\text{PBR}}(\boldsymbol{\beta}) := \|\tilde{V} - \Pi T^{\pi}\tilde{V}\|_{\mathbf{S}}^2$, where $\mathbf{S}$ is a diagonal matrix whose diagonal entries represent the stationary distribution of states in the input data, and $\Pi = \boldsymbol{\Phi}(\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\top}$ [1] is the orthogonal projector onto the feature space. The norm $\|\mathbf{x}\|_{\mathbf{S}} := \sqrt{\mathbf{x}^{\top}\mathbf{S}\mathbf{x}}$. With linear value function approximation it can be shown that $L_{\text{PBR}}(\boldsymbol{\beta}) = \mathbb{E}[\delta(\boldsymbol{\beta})\boldsymbol{\phi}(X)]^{\top}\mathbb{E}[\boldsymbol{\phi}(X)\boldsymbol{\phi}(X)^{\top}]^{-1}\mathbb{E}[\delta(\boldsymbol{\beta})\boldsymbol{\phi}(X)] = \|\mathbf{A}\boldsymbol{\beta} - \mathbf{b}\|_{\mathbf{G}}^2$. The corresponding empirical loss $\hat{L_{\text{PBR}}}$ can be obtained by replacing $\mathbf{A}, \mathbf{b}, \mathbf{G}$ with $\tilde{\mathbf{A}}, \tilde{\mathbf{b}}, \tilde{\mathbf{G}}$: $\hat{L_{\text{PBR}}}(\boldsymbol{\beta}) := \|\tilde{\mathbf{A}}\boldsymbol{\beta} - \tilde{\mathbf{b}}\|_{n\tilde{\mathbf{G}}}^2$, and the sample based feature space projector is given by $\hat{\Pi} = \tilde{\boldsymbol{\Phi}}(\tilde{\boldsymbol{\Phi}}^{\top}\tilde{\boldsymbol{\Phi}})^{-1}\tilde{\boldsymbol{\Phi}}^{\top}$.

We now discuss two different formulations of sparse TD learning through the l_1-regularized empirical PBR minimization.

3.1. Constrained Formulations

We first propose a constrained formulation where we have explicit control of the PBR:

$$\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{\beta}\|_1 \mid L_{\text{PBR}}(\boldsymbol{\beta}) \leq \epsilon^2 \right\}. \tag{9}$$

In the offline setting with finite samples, we solve the empirical version of the above problem

$$\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{\beta}\|_1 \mid \|\tilde{\mathbf{A}}\boldsymbol{\beta} - \tilde{\mathbf{b}}\|_{n\tilde{\mathbf{G}}} \leq \epsilon \right\}. \tag{10}$$

When $K > n$ and $\epsilon = 0$, it is a special case of finding the sparsest solution that satisfies the projected Bellman fixed-point equation and is also known as the *basis pursuit* problem.

We define $\tilde{\mathbf{d}} := \hat{\Pi}\tilde{R}, \tilde{\mathbf{C}} := \gamma\hat{\Pi}\tilde{\boldsymbol{\Phi}}' - \tilde{\boldsymbol{\Phi}}$ as in (Geist & Scherrer, 2012) and reformulate (10) into

$$\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{\beta}\|_1 \mid \|\tilde{\mathbf{d}} + \tilde{\mathbf{C}}\boldsymbol{\beta}\| \leq \epsilon \right\}. \tag{11}$$

The empirical PBR loss term used in (11) is equivalent to that used in (10) since

$$\|\tilde{\mathbf{C}}\boldsymbol{\beta} + \tilde{\mathbf{d}}\|^2 = \|\hat{\Pi}(\tilde{R} + \gamma\tilde{\boldsymbol{\Phi}}'\boldsymbol{\beta} - \tilde{\boldsymbol{\Phi}}\boldsymbol{\beta})\|^2 = \|\tilde{\mathbf{b}} - \tilde{\mathbf{A}}\boldsymbol{\beta}\|_{n\tilde{\mathbf{G}}}^2.$$

[1] When the Grammian matrix is not invertible, we can use its Moore-Penrose pseudoinverse.

Note that (11) is exactly the *basis pursuit denoising* (BPDN) problem. It finds the most sparse solution that satisfies the given tolerance on the PBR. The Euclidean projection onto the l_2 ball can be computed in closed-form, facilitating the development of a fast optimization algorithm.

3.2. Lagrangian Formulation

We consider the Lagrangian formulation of problem (9):

$$\min_{\boldsymbol{\beta}} \ \frac{1}{2}L_{\text{PBR}} + \rho\|\boldsymbol{\beta}\|_1. \tag{12}$$

Existing works have focused on offline algorithms for solving the empirical version of this problem with L_{PBR} replaced by $\hat{L}_{\text{PBR}}$. The empirical version can be treated as a vanilla unconstrained Lasso problem, so that many existing solvers can be directly applied, including LARS (Efron et al., 2004), FISTA (Beck & Teboulle, 2009), and FPC-BB (Hale et al., 2008). However, the stochastic version (12) is more challenging to solve, as also mentioned in (Liu et al., 2012). In Section 4.2, we propose a novel online algorithm for solving the stochastic optimization problem (12).

3.3. Extension to the Q-Function

It is straightforward to extend the above algorithms to estimate the state-action value function (Q-function) and integrate them into the policy iterations by following (Lagoudakis & Parr, 2003). The feature vector $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^K$ is augmented to $\boldsymbol{\phi}(\mathbf{x}, u) \in \mathbb{R}^{K\times|\mathcal{U}|}$. The definitions of $\tilde{\boldsymbol{\Phi}}$ and $\tilde{\boldsymbol{\Phi}}'$ are modified accordingly to incorporate the additional argument u. The Bellman equation, which Q^{π} satisfies, is $Q^{\pi}(\mathbf{x}, u) = r(\mathbf{x}, u) + \gamma\sum_{\mathbf{y}\in\mathcal{X}}\mathcal{P}(\mathbf{x}, u, \mathbf{y})Q^{\pi}(\mathbf{y}, \pi(\mathbf{y})), \quad \mathbf{x} \in \mathcal{X}$.

4. Learning Algorithms

Our approaches for solving the constrained formulations (BPDN (11)) is based on ADMM, which has been applied extensively in solving machine learning problems. For the Lagrangian formulation (12), we develop a specialized stochastic gradient descent method which is based on two approximation sequences of different speed.

4.1. BPDN (Offline)

We first explain an inexact version of the ADMM (Yang & Zhang, 2011), which was originally developed for solving compressed sensing problems. We apply variable-splitting to (11) and reformulate the problem into

$$\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{\beta}\|_1 + \mathcal{I}(\|\boldsymbol{\alpha}\| \leq \epsilon) \mid \tilde{\mathbf{d}} + \tilde{\mathbf{C}}\boldsymbol{\beta} = \boldsymbol{\alpha} \right\}. \tag{13}$$

The augmented Lagrangian of this constrained problem is $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{v}) = \|\boldsymbol{\beta}\|_1 + \mathcal{I}(\|\boldsymbol{\alpha}\| \leq \epsilon) - \mathbf{v}^{\top}(\tilde{\mathbf{d}} + \tilde{\mathbf{C}}\boldsymbol{\beta} - \boldsymbol{\alpha}) + \frac{1}{2\mu}\|\tilde{\mathbf{d}} + \tilde{\mathbf{C}}\boldsymbol{\beta} - \boldsymbol{\alpha}\|^2$, where $\mathbf{v}$ is the Lagrange multipliers.

For solving for $\boldsymbol{\alpha}$, we minimize $\min_{\boldsymbol{\alpha}} \ \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{v})$, which is equivalent to the following Euclidean projection problem onto the L_2 ball,

$$\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\|\boldsymbol{\alpha} - \mathbf{c}\|^2 \mid \|\boldsymbol{\alpha}\| \leq \epsilon \right\}, \tag{14}$$

where $\mathbf{c} = \tilde{\mathbf{d}} + \tilde{\mathbf{C}}\boldsymbol{\beta} - \mu\mathbf{v}$. The solution to the above projection problem is given by

$$\boldsymbol{\alpha} = \begin{cases} \mathbf{c}, & \text{if} \|\mathbf{c}\| \leq \epsilon \\ \epsilon \frac{\mathbf{c}}{\|\mathbf{c}\|}, & \text{otherwise.} \end{cases} \tag{15}$$

The subproblem w.r.t. $\boldsymbol{\beta}$ is

$$\min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\tilde{\mathbf{C}}\boldsymbol{\beta} + \tilde{\mathbf{d}} - \boldsymbol{\alpha} - \mu\mathbf{v}\|^2 + \mu\|\boldsymbol{\beta}\|_1. \tag{16}$$

Define $f(\boldsymbol{\beta}, \boldsymbol{\alpha}) := \frac{1}{2}\|\tilde{\mathbf{C}}\boldsymbol{\beta} + \tilde{\mathbf{d}} - \boldsymbol{\alpha} - \mu\mathbf{v}\|^2$, then $\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}, \boldsymbol{\alpha}) = \tilde{\mathbf{C}}^\top(\tilde{\mathbf{C}}\boldsymbol{\beta} + \tilde{\mathbf{d}} - \boldsymbol{\alpha} - \mu\mathbf{v})$, which is Lipschitz continuous with a Lipschitz constant $L = \lambda_{max}(\tilde{\mathbf{C}}^\top\tilde{\mathbf{C}})$, where λ_{max} denotes the largest eigenvalue. Problem (16) is a Lasso problem itself, and we can employ any Lasso solver as a subroutine. However, this step becomes very expensive in this way. Instead, we take a proximal gradient step to solve this subproblem approximately, which is much cheaper computationally. Specifically, given $\boldsymbol{\beta}_t$ and $\boldsymbol{\alpha}$, we solve

$$\min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\boldsymbol{\beta} - (\boldsymbol{\beta}_j - \tau\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}_j, \boldsymbol{\alpha}))\|^2 + \tau\mu\|\boldsymbol{\beta}\|_1, \tag{17}$$

which has a closed-form solution $\boldsymbol{\beta}_{j+1} = \mathcal{S}_{\tau\mu}(\boldsymbol{\beta}_j - \tau\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}_j, \boldsymbol{\alpha}))$, where $\mathcal{S}_\eta(\cdot) := \max(|\cdot| - \eta, 0)$ is the *shrinkage operator* (Tibshirani, 1996) with the shrinkage parameter η, and the operation is component-wise. It can be shown that ADMM with this proximal step converges if $\tau < \frac{1}{\lambda_{\max}(\tilde{\mathbf{C}}^\top\tilde{\mathbf{C}})}$. The stopping criteria that we use in the implementation is based on the primal and dual residuals (Boyd et al., 2010). A major advantage of this algorithm is that it is a first-order method, requiring only matrix-vector multiplications. The iterations are thus very fast to compute and simple to implement. We state the inexact ADMM algorithm in Algorithm 1.

Algorithm 1 ADMM-BPDN

Given $\tilde{\mathbf{C}}, \tilde{\mathbf{d}}, \mu, \tau, \epsilon$.
for $j = 0, 1, \cdots$ **do**
  $\mathbf{c} \leftarrow \tilde{\mathbf{d}} + \tilde{\mathbf{C}}\boldsymbol{\beta}_j - \mu\mathbf{v}_j$
  $\boldsymbol{\alpha}_{j+1} \leftarrow$ (15)
  $\boldsymbol{\beta}_{j+1} \leftarrow \mathcal{S}_{\tau\mu}(\boldsymbol{\beta}_j - \tau\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}_j, \boldsymbol{\alpha}_{j+1}))$
  $\mathbf{v}_{j+1} \leftarrow \mathbf{v}_j - \frac{1}{\mu}(\tilde{\mathbf{d}} + \tilde{\mathbf{C}}\boldsymbol{\beta}_{j+1} - \boldsymbol{\alpha}_{j+1})$
end for

4.2. L_1-regularized PBR minimization (Online)

We describe our online algorithm for solving the unconstrained stochastic optimization problem (12). The algorithm is inspired by the regularized dual averaging (RDA) algorithm (Xiao, 2010) with a specialized stochastic approximation of the L_{PBR} gradient. The gradient of $L_{PBR}(\boldsymbol{\beta})$ is $\nabla L_{PBR}(\boldsymbol{\beta}) = \mathbb{E}[(\boldsymbol{\phi}(X) - \gamma\boldsymbol{\phi}(Y))\boldsymbol{\phi}(X)^\top]\mathbb{E}[\boldsymbol{\phi}(X)\boldsymbol{\phi}(X)^\top]^{-1}\mathbb{E}[\delta(\boldsymbol{\beta})\boldsymbol{\phi}(X)]$. Note that the finite sample approximation $\hat{L_{PBR}}$ used in BPDN is a biased estimate of $L_{PBR}(\boldsymbol{\beta})$. Likewise, the direct finite sample approximation of the gradient is also biased without double sampling the next state Y. To get around this problem, we adopt the approach of (Sutton et al., 2009) for GTD2 and use an auxiliary variable $\boldsymbol{\omega}$ to approximate $\mathbb{E}[\boldsymbol{\phi}(X)\boldsymbol{\phi}(X)^\top]^{-1}\mathbb{E}[\delta(\boldsymbol{\beta})\boldsymbol{\phi}(X)]$. The gradient can now be estimated by sample approximation $\tilde{\nabla f_t}(\boldsymbol{\beta}) = \boldsymbol{\Delta}_t(\boldsymbol{\phi}_t^\top\boldsymbol{\omega}_t)$, where $\boldsymbol{\Delta}_t = \gamma\boldsymbol{\phi}'_{t+1} - \boldsymbol{\phi}_t$. The auxiliary variable $\boldsymbol{\omega}$ is updated by Least Mean Square (LMS) rule as in (Sutton et al., 2009). Each iteration of our algorithm then solves the following problem

$$\min_{\boldsymbol{\beta}} \langle \bar{\nabla f_t}, \boldsymbol{\beta}\rangle + \rho\|\boldsymbol{\beta}\|_1 + \frac{c}{2\sqrt{t}}\|\boldsymbol{\beta}\|^2$$

where $\bar{\nabla f_t} = \frac{1}{t}\sum_{s=1}^{t}\tilde{\nabla f_t}(\boldsymbol{\beta}^{(s)})$ is the time average of the gradient approximations, and c is a positive constant. Note that the coefficient for $\|\boldsymbol{\beta}\|^2$ above corresponds to the scaling factor $L_t = \frac{c\sqrt{t}}{2}$ as defined in (Xiao, 2010). The solution to this problem is $\boldsymbol{\beta}^{(t+1)} = \mathcal{S}_{\frac{\sqrt{t}\rho}{c}}\left(-\frac{\sqrt{t}}{c}\bar{\nabla f_t}\right)$.

We state the RDA algorithm with the customized gradient approximation in Algorithm 2. The computational and storage complexity of each iteration is linear $\mathcal{O}(K)$.

Algorithm 2 RDA with LMS update

Given $\boldsymbol{\omega}_0, \gamma, c, \{\eta_t\}, \bar{\nabla f_0} = 0$.
for $t = 1, \cdots, T_{\max}$ **do**
  $\boldsymbol{\Delta}_t \leftarrow \gamma\boldsymbol{\phi}'_{t+1} - \boldsymbol{\phi}_t$
  $\tilde{\nabla f_t} \leftarrow \boldsymbol{\Delta}_t(\boldsymbol{\phi}_t^\top\boldsymbol{\omega}_{t-1})$
  $\bar{\nabla f_t} \leftarrow \frac{1}{t}((t-1)\bar{\nabla f_{t-1}} + \tilde{\nabla f_t})$
  $\boldsymbol{\omega}_t \leftarrow \boldsymbol{\omega}_{t-1} + \eta_t(\delta_{t+1}(\boldsymbol{\beta}_{t-1}) - \boldsymbol{\phi}_t^\top\boldsymbol{\omega}_{t-1})\boldsymbol{\phi}_t$
  $\boldsymbol{\beta}_t \leftarrow \frac{\sqrt{t}}{c}\mathcal{S}_\rho(-\bar{\nabla f_t})$
end for

4.3. Convergence Analysis

Both of the proposed algorithms enjoy guaranteed convergence properties. We establish the convergence results in Theorems 4.1 and 4.2 below.

4.3.1. BPDN-ADMM (OFFLINE)

Theorem 4.1. *For any positive $\tau < \frac{1}{\lambda_{\max}(\tilde{\mathbf{C}}^\top\tilde{\mathbf{C}})}$, the sequence $\{(\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j, \mathbf{v}_j)\}$ generated by Algorithm 1 converges*

to an optimal solution to problem (11) *from any starting point.*

The proof parallels the one given in (Yang & Zhang, 2011) for compressed sensing and the one in (Ma et al., 2012) for latent variable Gaussian graphical model selection. The major steps are outlined in Appendix A the supplementary file.

4.3.2. RDA-LMS (ONLINE)

The original convergence results for RDA in (Xiao, 2010) no longer applies due to the fact that we are only approximating the gradients through maintaining a quasi-stationary sequence of $\boldsymbol{\omega}_t$. Our proof of convergence is based on the analysis of a two-time-scale stochastic approximation (Borkar, 1997) similar to that in (Sutton et al., 2009).

For notational conciseness, we use $\bar{\mathbf{g}}_t$ in place of $\bar{\nabla f_t}$ for the time-averaged gradient. The iterations in Algorithm 2 can be written as

$$\begin{cases} \bar{\mathbf{g}}_{t+1} = \frac{t}{t+1}\bar{\mathbf{g}}_t + \frac{1}{t+1}(\gamma\boldsymbol{\phi}'_{t+1} - \boldsymbol{\phi}_t)(\boldsymbol{\phi}_t^\top \boldsymbol{\omega}_t) \\ \boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \eta_t(\delta_{t+1}(\boldsymbol{\beta}_t) - \boldsymbol{\phi}_t^\top \boldsymbol{\omega}_t)\boldsymbol{\phi}_t \\ \boldsymbol{\beta}_{t+1} = q\mathcal{S}_\rho(-\bar{\mathbf{g}}_{t+1}). \end{cases}$$

Note that a different scalar is used in the update for $\boldsymbol{\beta}_t$. The original update in Algorithm 2 corresponds to using a scaling factor $c\sqrt{t}$ as suggested in (Xiao, 2010). For the convergence analysis here, we choose a scaling factor qt ($q > 0$) so that the update for $\boldsymbol{\beta}_t$ does not explicitly depends on t. Note that the sequence $\{qt\}$ is still non-negative and non-decreasing, as required in (Xiao, 2010). The above iteration can be further consolidated into

$$\begin{cases} \bar{\mathbf{g}}_{t+1} = \bar{\mathbf{g}}_t + \frac{1}{t+1}(-\bar{\mathbf{g}}_t + \hat{\mathbf{D}}_t\boldsymbol{\omega}_t) \\ \boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \eta_t\Big((\hat{\mathbf{b}}_t - q\hat{\mathbf{A}}_t\mathcal{S}_\rho(-\bar{\mathbf{g}}_t)) - \boldsymbol{\phi}\boldsymbol{\phi}^\top\boldsymbol{\omega}_t\Big), \end{cases} \tag{18}$$

where $\hat{\mathbf{b}}_t = \boldsymbol{\phi}_t r_{t+1}, \hat{\mathbf{A}}_t = \boldsymbol{\phi}_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}'_{t+1})^\top, \hat{\mathbf{D}}_t = (\gamma\boldsymbol{\phi}'_{t+1} - \boldsymbol{\phi}_t)\boldsymbol{\phi}_t^\top$, so that $\mathbf{b} = \mathbb{E}[\hat{\mathbf{b}}_t], \mathbf{A} = \mathbb{E}[\hat{\mathbf{A}}_t]$ by our definitions, and we define $\mathbf{D} := \mathbb{E}[\hat{\mathbf{D}}_t]$. We also define the following functions and variables:

$$\begin{aligned} F(\bar{\mathbf{g}}_t, \boldsymbol{\omega}_t) &:= -\bar{\mathbf{g}}_t + \mathbf{D}\boldsymbol{\omega}_t \\ M_{t+1} &:= (\hat{\mathbf{D}}_t - \mathbf{D})\boldsymbol{\omega}_t \\ \mathbf{K} &:= \mathbb{E}[\boldsymbol{\phi}\boldsymbol{\phi}^\top] \\ G(\bar{\mathbf{g}}_t, \boldsymbol{\omega}_t) &:= (\mathbf{b} - q\mathbf{A}\mathcal{S}_\rho(-\bar{\mathbf{g}}_t)) - \mathbf{K}\boldsymbol{\omega}_t \\ N_{t+1} &:= (\hat{\mathbf{b}}_t - \mathbf{b}) - q(\hat{\mathbf{A}} - \mathbf{A})\mathcal{S}_\rho(-\bar{\mathbf{g}}_t) - \\ &\quad (\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top - K)\boldsymbol{\omega}_t \end{aligned}$$

Now, the iteration (18) is in the form of a coupled stochastic recursion

$$\bar{\mathbf{g}}_{t+1} = \bar{\mathbf{g}}_t + \frac{1}{t+1}\big(F(\bar{\mathbf{g}}_t, \boldsymbol{\omega}_t) + M_{t+1}\big), \tag{19}$$

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \eta_t\Big(G(\bar{\mathbf{g}}_t, \boldsymbol{\omega}_t) + N_{t+1}\Big). \tag{20}$$

We prove the convergence of this coupled recursion through the following propositions and lemmas, which facilitate the application of Theorem 1.1 in (Borkar, 1997).

In all subsequent results, we make the following standard assumptions: (A1) The MDP is finite and stationary. (A2) The transition samples $\{(\boldsymbol{\phi}_t, r_{t+1}, \boldsymbol{\phi}'_{t+1})\}$ is an i.i.d. sequence with uniformly bounded second moments. (A3) The features defined by $\boldsymbol{\phi}$ are linearly independent, and $\mathbb{E}[\boldsymbol{\phi}\boldsymbol{\phi}^\top]^{-1}$ exists. (A4) The iterates $\{\bar{\mathbf{g}}_t, \boldsymbol{\omega}_t\}$ are bounded, i.e. $\sup_t \|\bar{\mathbf{g}}_t\| < \infty$, $\sup_t \|\boldsymbol{\omega}_t\| < \infty$.

Proposition 4.1. *The functions F and G are Lipschitz continuous.*

Proof. Since F is linear in both $\bar{\mathbf{g}}_t$ and $\boldsymbol{\omega}_t$, it is clearly Lipschitz continuous. For the Lipschitz continuity of G, it suffices to show that the shrinkage operator $\mathcal{S}_\rho(\cdot)$ is Lipschitz continuous. But it is known that $\mathcal{S}_\rho(\cdot)$ is non-expansive (Hale et al., 2008), and hence, it is Lipschitz continuous. □

Proposition 4.2. *For each $\bar{\mathbf{g}} \in \mathbb{R}^K$, the o.d.e.*

$$\dot{\boldsymbol{\omega}}_t = G(\bar{\mathbf{g}}, \boldsymbol{\omega}_t) \tag{21}$$

has a unique global asymptotically stable equilibrium $\lambda(\bar{\mathbf{g}})$ such that the function $\lambda(\cdot)$ is Lipschitz continuous.

Proof. With $\bar{\mathbf{g}}$ fixed, $G(\bar{\mathbf{g}}, \boldsymbol{\omega}_t) = (\mathbf{b} - q\mathbf{A}\mathcal{S}_\rho(-\bar{\mathbf{g}})) - \mathbf{K}\boldsymbol{\omega}_t$, where the first term is a constant. Since the features are linearly independent, $\mathbf{K}$ is symmetric positive definite. Hence, the o.d.e. (21) has a unique global asymptotic stable equilibrium $\mathbf{K}^{-1}(\mathbf{b} - q\mathbf{A}\mathcal{S}_\rho(-\bar{\mathbf{g}})) = \lambda(\bar{\mathbf{g}})$. Now, since $\mathcal{S}_\rho(\cdot)$ is non-expansive, the function $\mathbf{K}^{-1}\mathbf{A}\mathcal{S}_\rho$ is clearly Lipschitz continuous. Hence, the claim holds. □

Lemma 4.1. *For any $0 < q < \frac{1}{\sigma_{\max}(\mathbf{D}\mathbf{K}^{-1}\mathbf{A})}$, the o.d.e.*

$$\dot{\bar{\mathbf{g}}}_t = F(\bar{\mathbf{g}}_t, \lambda(\bar{\mathbf{g}}_t)) \tag{22}$$

has a unique global asymptotically stable equilibrium $\bar{\mathbf{g}}^*$.

Proof. We can rewrite the right-hand-side of the o.d.e. (22) as

$$F(\bar{\mathbf{g}}_t, \lambda(\bar{\mathbf{g}}_t)) = H(\bar{\mathbf{g}}_t) - \bar{\mathbf{g}}_t,$$

where $H(\bar{\mathbf{g}}_t) = \mathbf{D}\mathbf{K}^{-1}(\mathbf{b} - q\mathbf{A}\mathcal{S}_\rho(-\bar{\mathbf{g}}_t))$. The shrinkage operator $\mathcal{S}_\rho(\cdot)$ is non-expansive w.r.t. any l_p-norm, with $p \geq 0$ (Hale et al., 2008). Then, it is easy to see that for

any $0 < q < \frac{1}{\sigma_{\max}(\mathbf{D}\mathbf{K}^{-1}\mathbf{A})}$, $H(\cdot)$ is a contraction w.r.t. the l_2-norm. By Theorem 3.1 in (Borkar & Soumyanatha, 1997), the o.d.e. (22) is globally asymptotically stable. The equilibrium point $\bar{\mathbf{g}}^*$ satisfies

$$\begin{aligned} & F(\bar{\mathbf{g}}^*, \lambda(\bar{\mathbf{g}}^*)) = 0 \\ \Leftrightarrow \quad & \bar{\mathbf{g}}^* = H(\bar{\mathbf{g}}^*). \end{aligned}$$

Hence, $\bar{\mathbf{g}}^*$ is a fixed-point of the operator $H(\cdot)$. Since $H(\cdot)$ is a contraction w.r.t. the l_2-norm, the fixed-point is unique. □

Lemma 4.2. *The unique optimal solution to the PBR-Lasso problem*

$$\frac{1}{2} L_{PBR}(\boldsymbol{\beta}) + \rho \|\boldsymbol{\beta}\|_1. \tag{23}$$

is given by $\boldsymbol{\beta}^* = q\mathcal{S}_\rho(-\bar{\mathbf{g}}^*)$.

Proof. First, we observe that for any $0 < \xi_t < 1$, the iterate $\bar{\mathbf{g}}_{t+1} = \bar{\mathbf{g}}_t + \xi_t(-\bar{\mathbf{g}}_t + H(\bar{\mathbf{g}}_t))$ has the same fixed-point $\bar{\mathbf{g}}^*$ if it converges. In particular, with $\xi_t = \frac{1}{t+1}$, the above iterate is equivalent to

$$\bar{\mathbf{g}}_{t+1} = \frac{t}{t+1}\bar{\mathbf{g}}_t + \frac{1}{t+1} H(\bar{\mathbf{g}}_t), \tag{24}$$

which is exactly the time-averaged gradient of the loss function $L_{\text{PBR}(\cdot)}$ evaluated at $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_t$. By construction, the sequence $\{\boldsymbol{\beta}_t = q\mathcal{S}_\rho(-\bar{\mathbf{g}}_t)\}$ is the same as the one generated by the RDA algorithm (Xiao, 2010) applied to the strictly convex optimization problem (23) using the true gradients. By the convergence results in (Xiao, 2010), the sequence $\{\boldsymbol{\beta}_t\}$ converges to the unique optimal solution of problem (23). Hence, $q\mathcal{S}_\rho(-\bar{\mathbf{g}}^*) \equiv \boldsymbol{\beta}^*$, and the claim holds. □

Theorem 4.2. *The sequence* $\{\boldsymbol{\beta}_t, \boldsymbol{\omega}_t\}$ *generated by Algorithm 2 converges to* $(\boldsymbol{\beta}^*, \boldsymbol{\omega}^*)$ *w.p. 1.*

Proof. In order to apply Theorem 1.1 in (Borkar, 1997), we need to verify the following conditions:

- The functions F and G are Lipschitz continuous.
- The ξ_t and η_t satisfy

$$\begin{aligned} & \sum_t \xi_t = \infty, \quad \sum_t \eta_t = \infty, \\ & \sum_t \xi_t^2 < \infty, \quad \sum_t \eta_t^2 < \infty, \\ & \xi_t = (o)(\eta_t). \end{aligned}$$

- The sequences $\{M_t\}$ and $\{N_t\}$ satisfy

$$\sum_t \xi_t M_t < \infty, \quad \sum_t \eta_t N_t < \infty.$$

- For each $\bar{\mathbf{g}} \in \mathbb{R}^K$, the o.d.e. (21) has a unique global asymptotically stable equilibrium $\lambda(\bar{\mathbf{g}})$ such that the function $\lambda(\cdot)$ is Lipschitz continuous.
- The o.d.e. (22) has a unique global asymptotically stable equilibrium $\bar{\mathbf{g}}^*$.

The step size of the recursion (19) ξ_t is equal to $\frac{1}{t+1}$ in our case. Then, $\frac{\xi_t}{\eta_t} \to 0$ as $t \to \infty$, thus satisfying the above conditions. It is also easy to see that both M_{t+1} and N_{t+1} are martingale differences with bounded second moments, by (A1), (A2), (A4), and the non-expansiveness of $\mathcal{S}_\rho(\cdot)$. The third condition can be ensured by applying martingale convergence theorem on the martingales $\{\sum_{t=1}^s \xi_t M_t\}_{s=1}^\infty$ and $\{\sum_{t=1}^s \eta_t N_t\}_{s=1}^\infty$. The conditions for Theorem 1.1 in (Borkar, 1997) are all satisfied, given Propositions 4.1 and 4.2 and Lemma 4.1. An immediate consequence is that the sequences $\{(\bar{\mathbf{g}}_t, \boldsymbol{\omega}_t)\}$ generated by Algorithm 2 converge to $(\bar{\mathbf{g}}^*, \boldsymbol{\omega}^*)$ w.p. 1, where $\boldsymbol{\omega}^* = \lambda(\bar{\mathbf{g}}^*)$. By Lemma 4.2, the iterates $\{\boldsymbol{\beta}_t\}$ converge to $\boldsymbol{\beta}^*$ w.p. 1. □

4.4. Related Work

There have been some recent work on the development of a sparse LSTD method. LARS-TD (Kolter & Ng, 2009; Ghavamzadeh et al., 2011) applies the l_1-regularization to the projector onto the space of representable linear functions and solves the problem through LARS. (Johns et al., 2010) formulates the L1 regularized linear fixed-point problem as a linear complementarity problem. (Painter-wakefield & Parr, 2012) proposes a greedy algorithm based on orthogonal matching pursuit. l_1-PBR (Geist & Scherrer, 2012) instead applies the l_1-regularization to the classical PBR minimization, and (Mahadevan & Liu, 2012) develops an on-policy algorithm based on mirror-descent. Unlike the above algorithms, our BPDN algorithm solves a constrained formulation of the sparse TD learning problem which explicitly controls the PBR. The closest work to our method is probably (Geist et al., 2012) which constrains $\|\tilde{\mathbf{A}}\boldsymbol{\beta} - \tilde{\mathbf{b}}\|_\infty$. Our RDA-LMS algorithm, on the other hand, is an off-policy online algorithm based on gradient TD (Sutton et al., 2009). The only other l_1-based online sparse TD method that we are aware of is RO-TD (Liu et al., 2012), which minimizes a different loss function, $\|\tilde{\mathbf{A}}\boldsymbol{\beta} - \tilde{\mathbf{b}}\|_2$, based on the slack in (6) similar to D-LSTD. Note that the PBR is a weighted squared-norm of the fixed-point slack, being equivalent to $\|\tilde{\mathbf{A}}\boldsymbol{\beta} - \tilde{\mathbf{b}}\|_2^2$ only when the feature bases are orthonormal ($\mathbf{G} = \mathbf{I}$), which is not true in general. This difference appears to contribute to the different convergence speeds that we have observed below in Section 5.2.

5. Experiments

5.1. BPDN

We compared the proposed ADMM algorithm for BPDN-based sparse reinforcement learning with LSTD and D-LSTD (Geist et al., 2012) on two test problems, which are described in the next two sections. D-LSTD was formulated as a linear program and solved by the Mosek LP solver. For both problems, we constructed features using RBF basis functions with centers on grids of different densities and widths, five polynomial basis functions, as well as irrelevant features (white noise) of different lengths. We show results on fit errors (w.r.t. the true value functions) for value function approximation over 20 independent runs and simulated reward for the policy obtained from policy iterations over 10 independent runs.

5.1.1. Chain Walk

This is the classical 20-state test example from (Lagoudakis & Parr, 2003). Two actions (left or right) are available to control the current state. Visiting state 1 and 20 yields a reward of 1 and zero reward for anywhere else. Either action has a success probability of 0.9. Failure of an action bring the current state to the opposite direction.

For approximating the optimal value function, we collected 1000 training samples off-policy by following a random policy for 100 episodes, 10 steps each. The samples were collected in the same way for policy iterations, and we used a fixed set of samples throughout the policy iterations. The value function of any given policy can be computed exactly for this problem, and the optimal policy is to go to the left if the current state is from 1 to 10, and to the right otherwise.

Figures 1 and 2 show the comparison among BPDN, D-LSTD, and LSTD on the approximation error (RMSE) of the Q value function and the simulated reward of the learned policies for two cases of irrelevant features. The whiskers of the boxplots extend to the most extreme data point which is no more than 1.5 times the interquartile range from the boxes. BPDN and D-LSTD are significantly better than LSTD on both metrics in the presence of irrelevant features. BPDN is competitive to D-LSTD on simulated reward while doing tangibly better on value approximation. The average CPU times for BPDN on one instance of value approximation are 11.4s for $N_{\text{noise}} = 500$ and 15.7s for $N_{\text{noise}} = 1000$, and those for D-LSTD are 6.6s and 31.7s respectively. It is clear that BPDN has better scalability, being a first-order method.

5.1.2. Inventory Control

This is a single-shop single-product inventory control problem. The inventory level at the end of the day is a continuous variable between 0 and 20. (Think of each unit as a pallet of the products.) Orders of new items have to be made in batches of five (i.e. five actions). Customer demand for each day follows the continuous uniform distribution $[0, 24]$. There is a one-time order overhead of \$2 for any positive orders. The product unit cost is \$2, and the selling price is \$2.5 per unit. Any n unsold items incur an inventory cost of $\$0.2n^{1.4}$. Any unmet customer demand incurs a penalty of \$0.5 per unit. The goal is to learn an optimal planning strategy to operate the store.

We collected two sets of off-policy training samples with sizes 1000 and 5000 respectively. The samples were collected from different numbers of episodes of 10 steps each. The policy used for value function approximation tests is one that fills up the inventory as much as possible every day. Since the state space is continuous, the value function cannot be solved exactly. We used Monte Carlo rollouts to compute the value of the test states and inital actions associated with the test policy.

Figures 3 present the comparison results on the fitting error of the Q value function and the simulated reward of the learned policies. Similar to the Chain Walk example, BPDN was able to learn a more accurate Q value function and a better policy than LSTD from the same number of samples. We were unable to test D-LSTD on this domain because the large problem size led to excessive memory consumption by the Mosek LP solver, which reported out-of-memory error. This again shows that our first-order

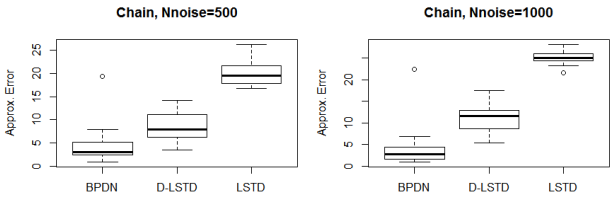

Figure 1. Chain Walk: Comparison on approximating the state-action value function of the optimal policy. The number of irrelevant features is 500 and 1000 in that order.

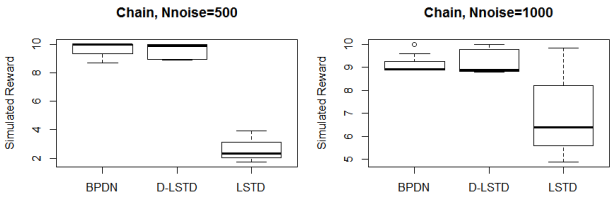

Figure 2. Chain Walk: Comparison on simulated reward from policy iterations. The number of irrelevant features is 500 and 1000 in that order.

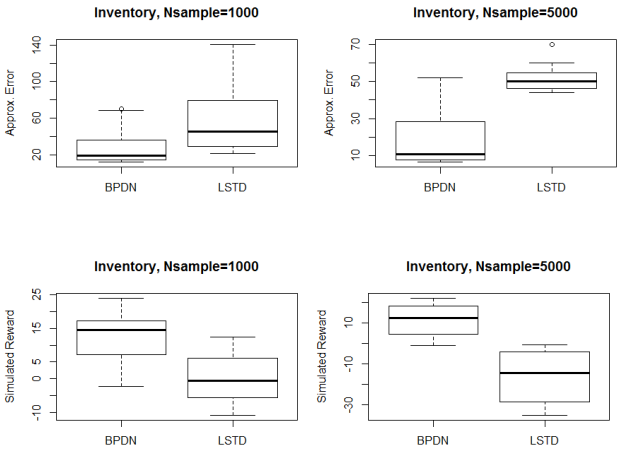

Figure 3. Inventory Control: Comparison on state-action value function approximation of an sub-optimal policy (top) and simulated reward from policy iterations (bottom). The number of irrelevant features is 1000. The number of training samples is 1000 for the left column and 5000 for right column.

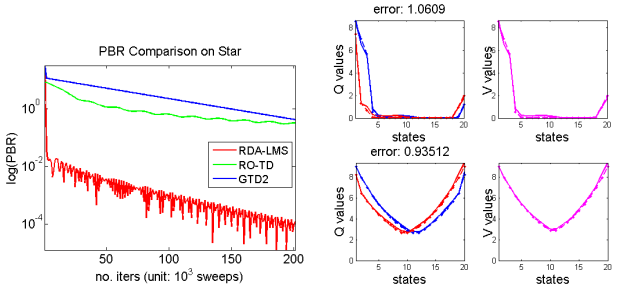

Figure 4. Left: Evolution of PBR on the Star example. Right: Q- and V-value functions approximation (by running RDA-LMS) associated with (bottom) the optimal policy and a suboptimal policy for Chain.

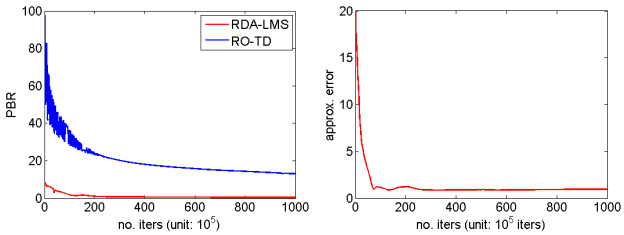

Figure 5. Evolution of PBR of RDA-LMS and RO-TD (left) and approximation error of RDA-LMS (right) for the Q-value function associated with the optimal policy for Chain.

method is more scalable and memory efficient.

5.2. RDA-LMS

We tested the RDA-LMS algorithm (Algorithm 2) on the 7-state Star MDP example (Sutton et al., 2009) and Chain domain to empirically verify the convergence of the algorithm. For the Star example, we set $\rho = 0$ to verify the convergence of the PBR to zero, and hence, the correctness of the algorithm. We plotted the evolution of the PBR for RDA-LMS, RO-TD and GTD2 in Figure 4. RDA-LMS was able to decrease the PBR much more rapidly than both RO-TD and GTD2. Comparisons were also made against RO-TD on Chain. We considered approximating the Q-value functions associated with the optimal policy and a sub-optimal policy. The transition samples were collected from 100 random episodes, with 10 consecutive steps each, totalling a collection of 1,000 samples. The algorithms swept through the samples multiple times. The features were constructed in the same way as in the previous section, except that there was no white noise features. Comparisons of the learned (solid lines) and true (dotted lines) value functions were plotted in the right-hand-side of Figure 4, which shows that the sparse solutions obtained by RDA-LMS approximate the true value functions faithfully. Figure 5 provides empirical evidence for the convergence of RDA-LMS with non-zero l_1-regularization. In addition, RDA-LMS decreased the PBR at a much faster rate than RO-TD.[2] We surmise that this is because RDA-LMS minimizes the PBR directly while RO-TD minimizes the slack in the fixed-point condition (6) as a proxy.

[2]The approximation error of RO-TD hovers above 20 and is not plotted.

6. Conclusion

In this paper, we have proposed two optimization algorithms for solving two different formulations of the sparse reinforcement learning problem: an ADMM-based off-line algorithm for constrained sparse TD learning and a novel online algorithm which combines elements from the RDA and LMS methods for unconstrained sparse TD learning. We provide convergence analysis for both algorithms, and promising preliminary test results have been reported in comparison to LSTD and two other sparse reinforcement learning algorithms. Future work will be focused on applications of the proposed sparse TD learning algorithms on real-world problems.

7. Acknowledgement

The authors would like to thank Niranjan Subrahmanya and Wendy (Lu) Xu for valuable discussions and the anonymous reviewers for helpful feedback. Special thanks to Bo Liu and Ji Liu for sharing the sample ROTD code.

References

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Bentivegna, Darrin C, Ude, Ales, Atkeson, Christopher G, and Cheng, Gordon. Humanoid robot learning and game playing using pc-based vision. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pp. 2449–2454. IEEE, 2002.

Borkar, V.S. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

Borkar, V.S. and Soumyanatha, K. An analog scheme for fixed point computation. i. theory. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 44(4): 351–355, 1997.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Machine Learning*, 3(1):1–123, 2010.

Bradtke, S.J. and Barto, A.G. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996.

Eckstein, J. and Bertsekas, D.P. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318, 1992. ISSN 0025-5610.

Efron, Bradley, Hastie, Trevor, Johnstone, Iain, and Tibshirani, Robert. Least angle regression. *The Annals of statistics*, 32(2): 407–499, 2004.

Geist, M. and Scherrer, B. l1-penalized projected bellman residual. *Recent Advances in Reinforcement Learning*, pp. 89–101, 2012.

Geist, M., Supélec, IMS, Metz, F., Scherrer, B., Nancy, F., Lazaric, A., and Ghavamzadeh, M. A dantzig selector approach to temporal difference learning. In *Proceedings of the 29th Annual International Conference on Machine Learning*, 2012.

Ghavamzadeh, M., Lazaric, A., Munos, R., and Hoffman, M. Finite-sample analysis of lasso-td. In *Proceedings of the 28th Annual International Conference on Machine Learning*, 2011.

Hale, E.T., Yin, W., and Zhang, Y. Fixed-Point Continuation for L1-Minimization: Methodology and Convergence. *SIAM Journal on Optimization*, 19:1107, 2008.

Johns, Jeffrey, Painter-Wakefield, Christopher, and Parr, Ronald. Linear complementarity for regularized policy evaluation and improvement. In *Advances in Neural Information Processing Systems*, pp. 1009–1017, 2010.

Kolter, J.Z. and Ng, A.Y. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 521–528. ACM, 2009.

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

Liu, Bo, Mahadevan, Sridhar, and Liu, Ji. Regularized off-policy td-learning. In *Advances in Neural Information Processing Systems 25*, pp. 845–853, 2012.

Ma, S., Xue, L., and Zou, H. Alternating direction methods for latent variable gaussian graphical model selection. Technical report, University of Minnesota, 2012.

Mahadevan, Sridhar and Liu, Bo. Sparse q-learning with mirror descent. In *UAI 2012*, 2012.

Ng, Andrew Y, Coates, Adam, Diel, Mark, Ganapathi, Varun, Schulte, Jamie, Tse, Ben, Berger, Eric, and Liang, Eric. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pp. 363–372. Springer, 2006.

Painter-wakefield, Christopher and Parr, Ronald. Greedy algorithms for sparse reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 1391–1398, 2012.

Proper, S. and Tadepalli, P. Scaling model-based average-reward reinforcement learning for product delivery. *Machine Learning: ECML 2006*, pp. 735–742, 2006.

Sutton, R.S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R.S., Maei, H.R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 993–1000. ACM, 2009.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

Tsitsiklis, J.N. and Van Roy, B. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1):59–94, 1996.

Tsitsiklis, J.N. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *Automatic Control, IEEE Transactions on*, 42(5):674–690, 1997.

Xiao, L. Dual averaging methods for regularized stochastic learning and online optimization. *The Journal of Machine Learning Research*, 11:2543–2596, 2010.

Yang, J. and Zhang, Y. Alternating direction algorithms for l1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.