Compressive Reinforcement Learning with Oblique Random Projections

Bo Liu and Sridhar Mahadevan
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{boliu, mahadeva}@cs.umass.edu

June 29, 2011

Abstract

Compressive sensing has been rapidly growing as a non-adaptive dimensionality reduction framework, wherein high-dimensional data is projected onto a randomly generated subspace. In this paper we explore a paradigm called *compressive reinforcement learning*, where approximately optimal policies are computed in a low-dimensional subspace generated from a high-dimensional feature space through random projections. We use the framework of oblique projections that unifies two popular methods to approximately solve MDPs – fixed point (FP) and Bellman residual (BR) methods, and derive error bounds on the quality of approximations obtained from combining random projections and oblique projections on a finite set of samples. We investigate the effectiveness of fixed point, Bellman residual, as well as hybrid least-squares methods in feature spaces generated by random projections. Finally, we present simulation results in various continuous MDPs, which show both gains in computation time and effectiveness in problems with large feature spaces and small sample sets.

1 Introduction

This paper explores a paradigm called *compressive reinforcement learning*, analogous to recent work on compressed sensing, wherein approximation spaces are constructed by

measurements representing random correlations with value functions. A random projection is a simple but elegant technique that has both a strong theoretical foundation and a wide range of applications including signal processing, medical image reconstruction, machine learning, data mining and so forth. Its theoretical foundation rests on the Johnson-Lindenstrauss lemma [16]. Briefly, given a set of samples S in a high-dimensional feature space R^n, if we construct an orthogonal projection of those sample points onto a random d-dimensional subspace, then if $d = O(\frac{\log|S|}{\epsilon^2})$, the projection is Lipschitz, that is, pairwise distances are preserved with high probability up to a distortion factor of $1 \pm \epsilon$. Intuitively, this process can be thought of as applying a spherically random rotation to a high-dimensional manifold and then reading off the first d coordinates [1]. Compared with other linear dimension reduction methods, like PCA (Principal Component Analysis), FA (Factor Analysis), etc, random projections are data-independent, which significantly reduces the computational cost of both representation discovery and control learning.

Two popular least-squares approaches for approximately solving MDPs are fixed-point (FP) or temporal-difference (TD) methods, and Bellman residual (BR) methods. A unified view of BR and FP methods can be constructed using oblique projections [12]. Hybrid least-squares methods interpolate between FP and BR approaches [5], and can also be viewed as computing oblique projections. Lazaric et al. [8] recently developed a finite-sample analysis of fixed point methods and provides new error bounds, which is based on the former work of [10]. One of the key ideas behind such error bound analysis is the contraction mapping property of the Bellman backup operator T. For any nonexpansive projection Π, if $\hat{v} = \Pi T\hat{v}$, then $\|\Pi T\hat{v} - \Pi Tv\| \leq \gamma \|\hat{v} - v\|$, for some $0 \leq \gamma < 1$, a property used in the error analysis of FP methods. The advantage of the oblique projection viewpoint is that it enables extending the finite-sample analysis to BR, FP, and hybrid least-squares methods in a unified manner.

In this paper, we use the framework of oblique projections as a geometric framework for error bound analysis, and derive theoretical results showing the loss that results from combining random projections with oblique projections in solving MDPs. In particular, we derive error bounds showing the approximation error introduced as a result of compressing the original (high-dimensional) feature space using a combined oblique projector and random projections, extending the previous results in [3] and [8]. In addition, we empirically demonstrate the effectiveness of Least-squares Policy Iteration [7] (LSPI) with random projections using experiments on various benchmark domains with the specific problem setting wherein a relatively small sample size and overcomplete basis set creates a challenging task for most state-of-the-art reinforcement learning algorithms. We also provide a comparison with L_1 regularization methods, such as LARS-TD [6].

Here is a brief roadmap to the rest of our paper. We begin by providing some background on projections in MDPs, summarizing the state of the art. We describe oblique

projections and finite-sample analysis, and how they are useful in characterizing solution methods for MDPs. We then give a detailed error analysis, building on the work discussed previously. Subsequently, we report on some simple numerical experiments where several comparison studies are carried out to show the effectiveness of LSPI with random projection. Finally, several promising potential research directions are discussed in the concluding section.

2 Approximate Solutions of Markov Decision Processes

A *Markov Decision Process* (MDP) [11] 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, and a reward model R. A policy $\pi : S \rightarrow A$ is a deterministic mapping from states to actions. Associated with each policy π is a value function v^π, which is a fixed point of the Bellman equation: $v^\pi(s) = T^\pi(v^\pi(s)) = R^\pi(s) + \gamma \int P^\pi(dy|s)v^\pi(y)$, where $0 \leq \gamma < 1$ is a discount factor, and P^π is the state transition function under fixed policy π. In what follows, we often drop the dependence of v^π on π, for notational simplicity. When the set of states S is large, it is often necessary to approximate the value function v using a set of basis functions (e.g., polynomials, radial basis functions, wavelets etc.). In linear value function approximation, a value function is assumed to lie in the linear span of a basis function matrix Φ of dimension $|S| \times k$. Hence, $v \approx \hat{v} = \Phi w$.

One approach to approximately solving the Bellman equation is the fixed point (FP) solution, often referred to as the TD approach, which is defined as finding a set of weights w_{FP} such that

$$w_{FP} = \arg\min_w \|\Pi^\Phi T(\Phi w) - \Phi w\|^2_\xi$$

where Π^Φ is an orthogonal projection onto the column space of Φ given by $\Phi(\Phi'\Xi\Phi)^{-1}\Phi'\Xi$. Here, Ξ is a diagonal matrix whose entries are specified by a weighted norm $\|\cdot\|_\xi$ such that $\|f\|^2_\xi = \sum_s \xi(s)f^2(s) = f'\xi f$, where $\xi(\cdot)$ is a measure over the state space. Fixed point solutions, therefore, look for a fixed point of the *combined* projected back up operator $\Pi^\Phi T$. Bellman residual methods, in contrast, find a set of weights w_{BR} under which the difference between the approximated value function and the backed up approximated value function $T(\Phi w)$ is minimized. That is,

$$w_{BR} = \arg\min_w \|T(\Phi w) - \Phi w\|^2_\xi$$

Least-squares policy iteration (LSPI)[7] is a well-known reinforcement learning method that can be combined with either the FP or BR projection method to find the optimal (approximate) value function that lies in the linear space of value functions spanned by Φ. LSPI focuses on iteratively improving policies by applying LSTD to the problem of estimating the value function, representing policies implicitly, and uses approximate policy iteration to find close to optimal policies.

3 Oblique Projection

Recently, the framework of oblique projections has been proposed as a way of unifying fixed point and Bellman residual methods [12, 17]. The oblique projection tuple (Φ, X) is defined as follows:

Definition 1 [12]: The *Oblique Projection* Π_X^Φ is the projection defined as $\Pi_X^\Phi = \Phi(X'\Phi)^{-1}X'$, which specifies a projection orthogonal to $span(X)$ and onto $span(\Phi)$.

Lemma 1 [12] : The BR, FP and H_2 solutions [5] can be unified under the framework of oblique projections with two properties:

$$\hat{v} = \Pi_X^\Phi T\hat{v} = \Pi^\Phi_{(I-\gamma P)'X} v, where\ X = \Xi(I-\beta\gamma P)\Phi \tag{1}$$

(1): the fixed point solution $\hat{v}$ of equation $Y = \Pi_X^\Phi TY$.
(2): the oblique projection of v onto subspace $span(\Phi)$ orthogonal to $span((I-\gamma P)'X)$, i.e., $\hat{v} = \Pi^\Phi_{(I-\gamma P)'X} v$, where v is the solution of Bellman equation $Y = TY$, and $X = \Xi(I-\beta\gamma P)\Phi$, where β is the mixing coefficient in the hybrid H_2 projection method proposed in [5] which controls the direction of oblique projection. Setting $\beta = 0$ gives rise to the FP method, and $\beta = 1$ results in BR projection.

Remark: The two perspectives offer two ways to find the solution of oblique projection: either by solving the fixed point equation w.r.t $\Pi_X^\Phi T$, or by first finding the solution of the Bellman equation $Y = TY$, and then projecting v onto $span(\Phi)$ with oblique projection $\Pi^\Phi_{(I-\gamma P)'X}$, which will be used in Proposition 2 later.

Definition 2: $\Pi^\Phi v$ is defined as the "best" approximation of v onto subspace $span(\Phi)$, i.e., $\Pi^\Phi v = \underset{f \in span(\Phi)}{\arg\inf} \|v - f\|_\xi^2$, and the best approximation can be computed as:

$$\Pi^\Phi v = \Phi(\Phi'\Xi\Phi)^{-1}\Phi'\Xi(I-\gamma P)^{-1}R. \tag{2}$$

Lemma 2 [12]: For any choice of subspace X, the approximation error satisfies:

$$\left\|v - \Pi_X^\Phi v\right\|_\xi = \|\Pi_{L'X}\|_\xi \left\|v - \Pi^\Phi v\right\|_\xi = \sqrt{\sigma(G_1G_2)}\left\|v - \Pi^\Phi v\right\|_\xi.$$

Here, $\sigma(G_1G_2)$ is the spectral radius of product of two Gramian matrices, where $L = I - \gamma P$.

$$G_1 = (\sqrt{\Xi}\Phi)'(\sqrt{\Xi}\Phi), G_2 = (\sqrt{\Xi^{-1}}(L'X)(\Phi'L'X)^{-1})' \cdot (\sqrt{\Xi^{-1}}(L'X)(\Phi'L'X)^{-1}) \quad (3)$$

4 Finite-sample Analysis

A modified assumption of the martingale difference ε_t is proposed based on [8]. The difference between Assumption 1 and the one in [8] is that there are, per se, two filtrations defined here. The *adaptation filtration* is the whole trajectory $\{x_i\}_{i=1}^n$, which means that ε_t is adapted to the whole trajectory $\{x_i\}_{i=1}^n$; The *expectation filtration* is the past trajectory $\{x_i\}_{i=1}^t$, which means that the expectation should be centered conditioned on the filtration of only the past. Assumption 1 is required so that ε_t is a martingale difference sequence and thus Azuma's inequality can be applied to error bound analysis in Proposition 3.

Assumption 1: (Modified Markov Design Setting) The sample path $\{(x_t, y_t)\}_{t=1}^n$ is generated by a Markov chain, where the corresponding filtration is $\mathcal{F} = \{x_i\}_{i=1}^n$. The regression model is $y_t = f(x_t) + \varepsilon_t$, f is the target function, and the noise term ε_t is $\mathcal{F} - measurable$, and is bounded so that $|\varepsilon_t| \leq C$ and satisfies the martingale difference property $E[\varepsilon_t | x_1, \cdots, x_t] = 0$.

In [8], the definition of Pathwise Bellman Operator is introduced, which can be considered a core concept of the finite-sample analysis framework.

Definition 3: The *Pathwise Bellman Operator* [8] is computed based on the trajectory of a single path $\{(x_t, y_t)\}_{t=1}^n$, which yields the one-step "empirical" Bellman operator $\hat{T} : \Re^n \to \Re^n$:

$$\hat{T}y_t = r_t + \gamma \hat{P} y_t, 1 \leq t \leq n \quad (4)$$

$\hat{P} : \Re^n \to \Re^n$ is defined as the operator $\hat{P}_{i,j} = \mathcal{I}(j = i+1, i \neq n)$ such that $\hat{P}y_t = y_{t+1}, 1 \leq t < n$ and $\hat{P}y_n = 0$. In a single trajectory, $\hat{P}$ can be set as a nilpotent upper shift matrix [2, 15].

$$\hat{P} = \begin{bmatrix} \vec{\mathbf{0}}' & I_{n-1} \\ 0 & \vec{\mathbf{0}} \end{bmatrix}, \vec{\mathbf{0}} = \begin{bmatrix} 0, & \cdots, & 0 \end{bmatrix}_{1,n-1} \quad (5)$$

Likewise, the k-step ($k > 1$) Bellman backup operator $\hat{T}^k$ over sample x_t is derived in a similar way as in [14]:

$$\hat{T}^k(y_t) = (\hat{T})^k(y_t) = \begin{cases} r_t + \cdots + \gamma^k(y_{t+k}), & 1 \leq t < n-k \\ r_t + \cdots + \gamma^{n-t} r_n + \gamma^{n-t} y_n, & t \geq n-k \end{cases}$$

One nice property of the pathwise Bellman operator is that it is a γ-contraction mapping in l_2 norm.

Definition 4: Given the pathwise Bellman operator $\hat{T}$ and oblique projection Π_X^Φ, the *pathwise Bellman solution* is defined as the solution of the Bellman equation $Y = \hat{T}Y$, and the *pathwise LSPI solution* is defined as $Y = (\Pi_X^\Phi \hat{T})Y$.

Proposition 1: (Pathwise Bellman Solution) Given the pathwise Bellman operator $\hat{T}$ and the Bellman equation $Y = \hat{T}Y$, $(\hat{T})^n v$ is the solution of the equation.

Proof: First let us fix t and prove that for $\forall i > n - t, \hat{T}^i = \hat{T}^{n-t}$. According to the definition above, $\forall i > n - t, \hat{T}^i y_t = r_t + \cdots + \gamma^{n-t} r_n + \gamma^{n-t} y_n$, so we have $\forall t, \forall i > n, (\hat{T})^i = (\hat{T})^n$. Then we have $\hat{T}(\hat{T}^n v) = (\hat{T}^n v)$, which shows that $\hat{T}^n v$ is the solution of the Bellman equation. It is easy to prove that the true value function v does not satisfy this equation unless $\hat{T} = T$, so we have $\hat{T}v \neq v$ if $\hat{T} \neq T$. $\hat{T}^n v$ is the unique solution since $\hat{T}$ is a $\gamma-$contraction mapping in l_2 norm.

Remark: This means that given the biased pathwise empirical Bellman operator $\hat{T}$, what is finally learned is not the true value v, but the ***biased value*** $\hat{T}^n v$.

Now we can combine the Markov design bound with the multi-step Bellman backup operator.

Proposition 2: (Pathwise LSPI Solution) Given the Pathwise Bellman Operator $\hat{T}$, and oblique projection Π_X^Φ, the pathwise LSPI solution is $\Pi^\Phi_{(I-\gamma\hat{P})'X}(\hat{T}^n v)$.

Proof: Given the Pathwise Bellman Operator $\hat{T}$, solving the equation can be divided into two steps:

(1): Solve the pathwise Bellman solution of $Y = \hat{T}Y$. From Proposition 1, the solution is biased $\hat{T}^n v$.

(2): Project $\hat{T}^n v$ onto $span(\Phi)$ with oblique projector $\Pi^\Phi_{(I-\gamma\hat{P})'X}$, which is actually $\Pi^\Phi_{(I-\gamma\hat{P})'X}(\hat{T}^n v)$, where $X = \Xi(I - \beta\gamma\hat{P})\Phi$.

Proposition 3: Given the trajectory of the Markov chain satisfying Assumption 1, let v be a vector whose components are the true values at $\{x_i\}_{i=1}^n$. Then with probability $1-\delta$, we have

$$\left\| \Pi^\Phi \hat{T}^n v - \Pi^\Phi v \right\| \leq \gamma V_{\max} L \sqrt{\frac{d}{v_n}} \left(\sqrt{\frac{8}{n} \log\left(\frac{4d}{\delta}\right)} + \frac{1}{n} \right) \tag{6}$$

where v_n is the smallest strictly-positive eigenvalue of the sample-based Gramian matrix $\frac{1}{n}\Phi'\Phi$.

Proof: In our proof, to extend Assumption 1 to the reinforcement learning setting, the target function f is replaced by value function V, and ε_t is replaced by the multi-step temporal difference along eligible traces, i.e., $\varepsilon_t = (\hat{T}^n)(V(x_t)) - V(x_t)$. The value $V(x_t)$

at sample x_t w.r.t the true state transition kernel P can be represented as

$$\begin{aligned} V(x_t) &= r_t + \gamma \int P(dy|x_t)V(y) \quad (7) \\ &= r_t + \gamma r_{t+1} + \cdots + \gamma^k \int P^k(dy|x_t)V(y), 1 < k \leq n-t \end{aligned}$$

Now we prove ε_t is still a martingale difference sequence. Replace $y(t), y(n)$ by $V(x_t), V(x_n)$, respectively and introduce (7) to the definition of multi-step temporal difference, and set $k = n - t$

$$\begin{aligned} \varepsilon_t &= (\hat{T}^n)V(x_t) - V(x_t) \quad (8) \\ &= r_t + \gamma r_{t+1} + \cdots + \gamma^{n-t}V(x_n) - r_t - \gamma r_{t+1} - \cdots - \gamma^{n-t} \int P^{n-t}(dy|x_t)V(y) \\ &= \gamma^{n-t}\left[V(x_n) - \int P^{n-t}(dy|x_t)V(y)\right] \leq \gamma\left[V(x_n) - \int P^{n-t}(dy|x_t)V(y)\right], \\ &\quad 1 \leq t < n \quad (9) \end{aligned}$$

and $\varepsilon_n = -\gamma \int P(dy|x_t)V(y)$. Thus, ε_t is martingale difference noise whose expectation depends on $\{x_i\}_{i=1}^t$ and x_n. Since given the sample, information about x_n can be considered as a constant, thus ε_t satisfies $|\varepsilon_t| \leq 2\gamma V_{\max}, E[\varepsilon_t|x_1, \cdots, x_t] = 0$. Applying concentration inequality here, it follows (6) holds with probability $1 - \delta$. Details are not given here due to limited space, but the proof generally follows that given in [8].

5 Error Bound Analysis

In this section a new error bound analysis is given which extends the results in [3] from several perspectives. Firstly, results in [3] are limited to the fixed point/LSTD solution, whereas the analysis in our paper broadly extends to any oblique projection, including FP, BR and hybrid least square methods proposed in [5]. More importantly, we point out given the biased pathwise Bellman operator, the solution is not the true value v but a biased solution $\hat{T}^n v$. Here we introduce the notion of random space. Given a D-dimensional linear space F spanned by basis set Φ, i.e., $F = \{f|f(\cdot) = \Phi(\cdot)'\alpha, \alpha \in \Re^D\}$, where $\Phi(\cdot) = [\phi_1(\cdot), \phi_2(\cdot), \cdots, \phi_D(\cdot)]'$ is the feature vector. A $d-$dimensional random space G ($d < D$) is generated by $G = \{g|g(\cdot) = \Psi(\cdot)\beta, \beta \in \Re^d\}$, where the feature vector is $\Psi(\cdot) = [\psi_1(\cdot), \psi_2(\cdot), \cdots, \psi_d(\cdot)]'$ and $\Psi(\cdot) = C\Phi(\cdot)$ with $C = [c_{i,j}]_{d\times D}, c_{i,j} \sim N(0, \frac{1}{d})$. $K_F = \frac{1}{n}\Phi'\Phi, K_G = \frac{1}{n}C\Phi'\Phi C'$ are sample-based Gramian matrices w.r.t space F, G respectively. If the Markov chain admits a stationary distribution ρ, then the Gramian matrix

H, G can also be defined via ρ as

$$K_{Fi,j} = \int \phi_i(x)\phi_j(x)\rho(dx), K_{Gi,j} = \int \psi_i(x)\psi_j(x)\rho(dx)$$

v_n is denoted as the smallest eigenvalue of the sample-based Gramian K_G.

Lemma 3 [3]: For any vector v in the random projection space $g \in G$, if $d \geq 15\log(\frac{4n}{\delta})$, we have with probability $1-\delta$

$$\left\|\Pi^G v - \Pi^F v\right\| \leq \sqrt{\frac{8\log(\frac{8n}{\delta})}{d}} m(\Pi^F v), \tag{10}$$

where n is the number of samples, d is the dimension of compressed feature space, $m(f)$ is defined as $m(f) = \sup_{x\in X} \|\phi(x)\|_2$. With this background in place, we can now turn to characterizing our main theoretical results.

Theorem 1: Given value function v, finite sample set S, high-dimensional feature space F and low-dimensional feature space G generated by random projections, and the oblique projection Π_X^G orthogonal to $span(X)$ onto the random projection space $span(G)$, and the *pathwise LSPI solution* $\hat{v} = (\Pi_X^\Phi \hat{T})\hat{v}$

$$\begin{aligned} \|v - \hat{v}\| \quad \leq \quad & \left\|v - \Pi^F v\right\| + \sqrt{\frac{8\log(\frac{8n}{\delta})}{d}} m(\Pi^F v) \\ & + \gamma V_{\max} L \sqrt{\frac{d}{v_n}} \left(\sqrt{\frac{8\log(\frac{4d}{\delta})}{n}} + \frac{1}{n}\right) + \sqrt{\sigma(G_1 G_2) - 1}\left\|\hat{T}^n v - \Pi^G \hat{T}^n v\right\| \end{aligned} \tag{11}$$

Proof: Consider the illustration shown in Figure 1. First, the error between the true value v and the finite-sample based pathwise LSPI solution $\hat{v}$ is bounded by the other three sides of the triangle $(v, \hat{v}, \hat{T}^n v, \Pi^G \hat{T}^n v)$,

$$\|v - \hat{v}\| \leq \left\|v - \Pi^G v\right\| + \left\|\Pi^G v - \Pi^G \hat{T}^n v\right\| + \left\|\Pi^G \hat{T}^n v - \hat{v}\right\| \tag{12}$$

We have developed bounds on each of these three sides.
((1) $\left\|v - \Pi^G v\right\|$: From Lemma 3, we have:

$$\left\|v - \Pi^G v\right\| \leq \left\|v - \Pi^F v\right\| + \left\|\Pi^F v - \Pi^G v\right\| \leq \left\|v - \Pi^F v\right\| + \sqrt{\frac{8\log(\frac{8n}{\delta})}{d}} m(\Pi^F v) \tag{13}$$

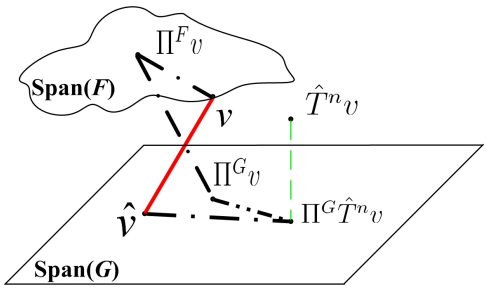

Figure 1: Finite-sample Analysis of Oblique Random Projections

(2) $\left\|\Pi^G v - \Pi^G \hat{T}^n v\right\|$: According to Proposition 3, $\left\|\Pi^G v - \Pi^G \hat{T}^n v\right\|$ is bounded as follows:

$$\left\|\Pi^G v - \Pi^G \hat{T}^n v\right\| \leq \gamma V_{\max} L \sqrt{\frac{d}{v_n}} \left(\sqrt{\frac{8\log(\frac{4d}{\delta})}{n}} + \frac{1}{n}\right) \tag{14}$$

(3) $\left\|\Pi^G \hat{T}^n v - \hat{v}\right\|$: Next, $\left\|\hat{T}^n v - \hat{v}\right\|$ is bounded with the results in Lemma 2:

$$\left\|\hat{T}^n v - \hat{v}\right\| \leq \sqrt{\sigma(G_1 G_2)} \left\|\hat{T}^n v - \Pi^G \hat{T}^n v\right\| \tag{15}$$

G_1, G_2 are defined as in (3), where X is defined in (1). The following inequality arises from using the Pythagorean theorem and the triangle inequality over the triple, which is $(\hat{v}, \Pi^G \hat{T}^n v, \hat{T}^n v)$:

$$\left\|\Pi^G \hat{T}^n v - \hat{v}\right\| \leq \sqrt{\sigma(G_1 G_2) - 1} \left\|\hat{T}^n v - \Pi^G \hat{T}^n v\right\| \tag{16}$$

Finally, from (12) to (16), (11) can be proved. In (11), the first term $\left\|v - \Pi^F v\right\|$ is the model-based error which only depends on the capacity of the function space F, namely, how well the function space F can approximate the value function v. The second term, which is called estimation error [3], is primarily generated by inherent "noise" due to sampling, which reflects the difference between the true Bellman backup operator T and the Pathwise Bellman operator $\hat{T}$. As the number of samples n increases, the estimation error will decrease to zero. The complexity of the model will also help increase the estimation error. Theorem 1 extends the results in [3] from the following perspectives. Firstly, the results in [8, 3] are limited to the fixed point/LSTD solution, whereas the analysis in our paper broadly extends to any solution generated by oblique projection.

The existence of v_n is a problem since the sample-based Gramian matrix $\frac{1}{n} C\Phi'\Phi C'$ may not be invertible. In [3], it is shown that given a large enough set of samples, the

smallest eigenvalue v_n of the sample-based Gramian K_G is strictly positive with high probability (w.h.p), and also w.h.p v_n is bigger than the smallest eigenvalue ω of the Gram matrix $\frac{1}{n}\Phi'\Phi$ of the high-dimensional space F. [3] also proved that the number of samples needed for the empirical Gram matrix K_G in subspace G to be nonsingular is less than that for its counterpart K_F in high-dimensional space F w.h.p under some conditions. Here we give an alternative way to understand this, which is much more explicit. The analysis relies on a result in [4], that is, in a high-dimensional space, there exists a much larger number of ***almost*** orthogonal than orthogonal directions.

Theorem 2: If the high-dimensional Gramian matrix $\frac{1}{n}\Phi'\Phi$ satisfies $rank(\frac{1}{n}\Phi'\Phi) \geq d$, the low-dimensional Gramian matrix $\frac{1}{n}C\Phi'\Phi C'$ will be nonsingular, i.e., $rank(\frac{1}{n}C\Phi'\Phi C') = d$, where C is the randomly generated compression matrix, and its smallest eigenvalue v_n is strictly positive.

Proof: First denote $K_F = \frac{1}{n}\Phi'\Phi$. If $rank(K) \geq d$, there exists a $d * d$ square sub-matrix K_{sub} of K such that $rank(K_{sub}) = d$. Next draw arbitrary d rows and columns from C, which forms $d * d$ square matrix C_{sub} such that

$$rank(C_{sub}K_{sub}C'_{sub}) \leq rank(CK_FC') \leq d \tag{17}$$

Since each column of C is approximately orthogonal, so is C_{sub}. Also since the rank of a matrix is invariant by left-multiplying a full column-rank matrix or right-multiplying a full row-rank matrix, we have $rank(C_{sub}K_{sub}C'_{sub}) = d$. So, from above two equations we have $rank(\frac{1}{n}C\Phi'\Phi C') = d$ and thus its smallest eigenvalue v_n is strictly positive.

6 Experiments

Finally, we report on some simple experiments on two benchmark domains to illustrate the theoretical analysis. Hybrid [5] of LSTD-RP and BR-RP is used at each iteration of the Compressed Reinforcement Learning algorithm. The algorithm description of LSTD-RP is identical to the one in [3]. The algorithm is sketched as follows,

Algorithm 1: General Framework of Compressive Reinforcement Learning

Input: $(D, d, \{x_t\}_{t=1}^{n}, \phi, \gamma, \beta)$

Compute:

- Compute high-dimensional matrix $\Phi_{n\times D}$
- Generate projection matrix $C = [c_{i,j}]_{d\times D}, c_{i,j} \sim N(0, \frac{1}{d})$
- Compute low-dim feature matrix $\Psi_{n\times d} = \Phi_{n\times D} C'$
- Compute $\hat{P}\Psi = \left[\Psi(x_2)'; \Psi(x_3)'; \cdots, \Psi(x_n)'; \vec{\mathbf{0}}'\right]'$
- Compute $A_{H_2} = (\Psi - \beta\gamma\hat{P}\Psi)'(\Psi - \gamma\hat{P}\Psi), b_{H_2} = (\Psi - \beta\gamma\hat{P}\Psi)'r$

Return: $w = A_{H_2}^{+} b_{H_2}$

The inverted pendulum [7] is a standard continuous-state MDP, where the state space is defined by the vertical angle θ and the angular velocity $\dot{\theta}$. The three discrete actions are applying a force of $-50, 0$, or 50 Newtons. In the experiment, a run is deemed successful if it can balance the pole for 500 steps. In our experiment, $D = 1200, d = 240, N = 1500$, i.e., 1200 RBF kernels are used in the high-dimensional space and this dimension is compressed to 240 in the low-dimensional space, and the kernels are generated from the collected samples via the k-means algorithm. Figure 2 shows comparison study of Compressive Reinforcement Learning(BR solution, i.e., hybrid factor $\beta = 1$), LARS-TD, and LSPI on number of episodes the pendulum can balance with the number of iterations based on average of 100 runs for each number of iteration. LARS-TD performs the best, whereas LSPI performs the worst.

Trial #	Noise Setting
1	5% uniform action noise
2	10% uniform action noise
3	Gaussian action noise $N(0, 0.1)$
4	Gaussian action noise $N(0, 0.2)$
5	5% uniform sensor noise
6	10% uniform sensor noise
7	Gaussian sensor noise $N(0, 0.1)$
8	Gaussian sensor noise $N(0, 0.2)$
9	Noise free

Table 1: Noise Trial Setting of Acrobot

In the Acrobot domain [9], the goal is to raise the tip of the second link of the Acrobot above a certain height in minimum time. The parameters are set as $D = 1200, d = 216$, 3000 samples are collected from 15 episodes with 250 steps in each episode. Figure 3 shows the comparison of computation time of the three methods. Compressive Reinforcement Learning's computation time is roughly 2.4 second per iteration, whereas that of LARS-TD and LSPI are around 12 and 18 seconds per iteration. Success of Compressive Reinforcement Learning is evident here as it remarkably reduces the computation time with very little burden of computation cost. Figure 4 and 5 shows a comparison of LARS-TD and Compressive Reinforcement Learning (FP solution, i.e., hybrid factor $\beta = 0$) over 50 runs. 4 shows the number of steps to swing up the Acrobot in terms of different number of iterations. When the number of iterations is 5, both methods failed. The average number of iterations for LARS-TD to converge is 14, and it takes around 174 steps to swing up when it converges, whereas that for Compressive Reinforcement Learning is 19 iterations and 189 steps with FP solution. 5 is simulation under various noise situations to test the robustness of the algorithm. Both action noise and sensor noise are added where action noise is added to change magnitude of the force and sensor noise is added on θ_1 of the Acrobot, i.e., the angle measure of the first arm, and the noise type is both Gaussian noise and Uniform noise. Gaussian noise $\varepsilon_G \sim N(0, \sigma)$ with zero mean and specified variance is added as $u = u + \varepsilon_G$, where u is either the action or θ_1 . Uniform noise $\varepsilon_U \sim U(0, 1)$ is added as $u = u * (1 + \varepsilon_U)$, similar as in [13]. There are 9 noise trial situations listed in Table 1, and the comparison result is shown in Figure 5.

The effectiveness of Compressive Reinforcement Learning, therefore, is clearly demonstrated by these experiments. First, when dealing with problems with large feature space, Compressive Reinforcement Learning can effectively reduce the dimension in an alternative way besides sparsity of the regression solution at the cost of very low extra computation with the advantage of a linear data-independent projection. Secondly, the computation time at each iteration is sharply reduced compared with ordinary LSPI, and is at the same level as LARS-TD. It is also noteworthy to mention that combining random projection for feature compression and LARS-TD for feature selection can improve the performance and reduce the computation time even further. The details of these additional experiments is not reported here due to space.

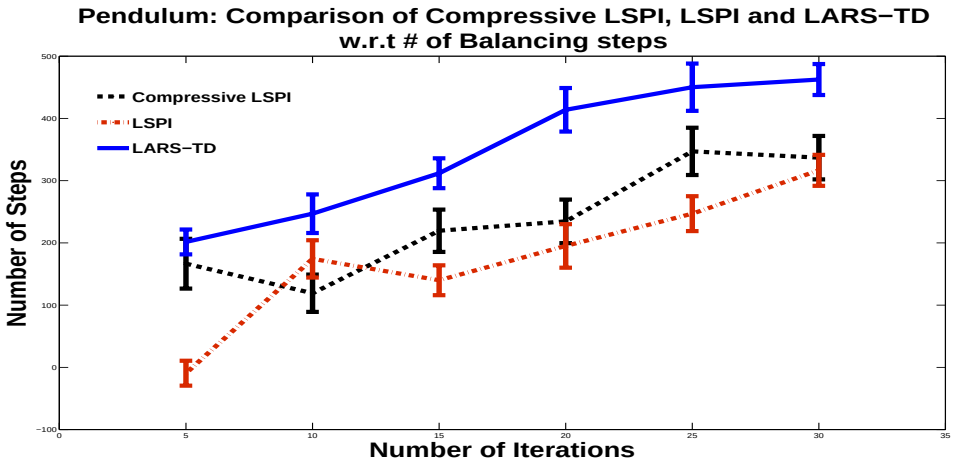

Figure 2: Pendulum: Comparison of Compressive Reinforcement Learning, LSPI and LARS-TD

7 Summary

In this paper we analyzed a framework called *compressive reinforcement learning*, which investigates the use of random projections in approximately solving Markov decision processes. Here, the approximation subspace is generated by a random projection of the original high-dimensional feature space. Compressed sensing is one of the most active areas in signal processing and our paper represents one of the first studies that combines both theoretical analysis and experiments. We used the framework of oblique projection to obtain error bounds that cover both fixed point and Bellman residual solutions, as well as hybrid least-squares methods. A novel error analysis, which integrates finite sample analysis, random projections and oblique projections, is derived. Finally, simulation results on benchmark MDP domains was provided to demonstrate the validity of the proposed approach. Overall, the proposed framework provides an interesting alternative to feature compression/selection and an alternative to L_1 penalization techniques such as LASSO.

Much remains to be explored in this new framework. On the experimental side, we need to evaluate the scalability of the proposed approach in larger domains. On the theoretical side, we intend to exploit bounds known for sample trajectories generated by stationary and non-stationary β-mixing Markov chains. In our approach, the designer can choose a random projection subspace of lower dimension and reduce the estimation error (sample-based error) at the cost of a controlled increase in model-based error. How to determine d of random projection is a key issue to be investigated further.

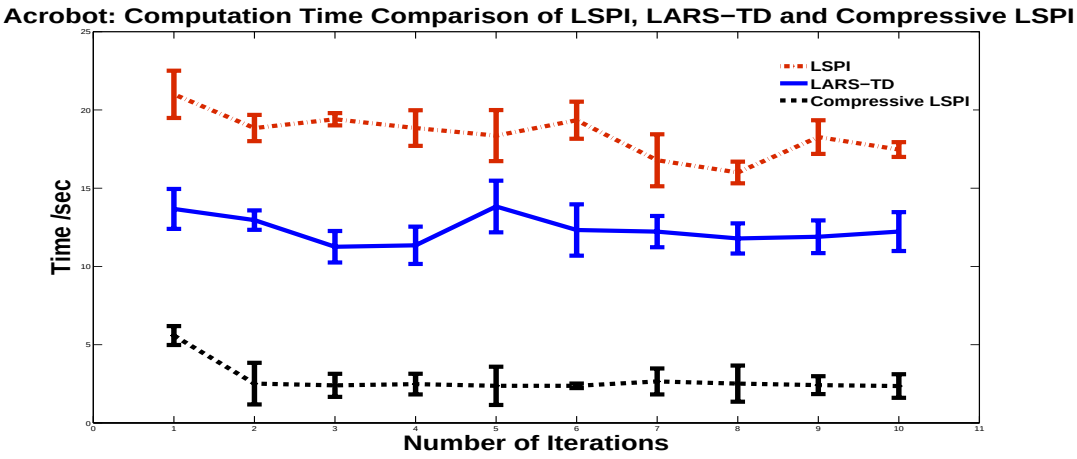

Figure 3: Acrobot: Computation Time Comparison of Compressive Reinforcement Learning, LSPI and LARS-TD

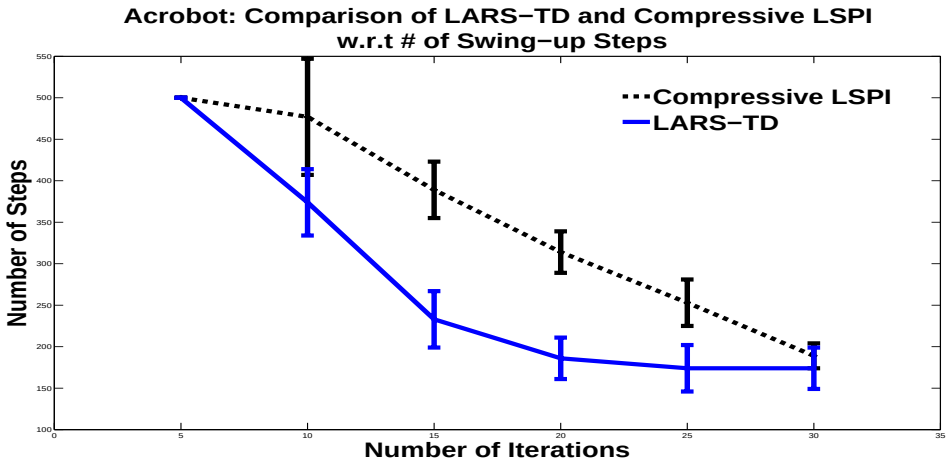

Figure 4: Acrobot: Comparison of Compressive Reinforcement Learning and LARS-TD

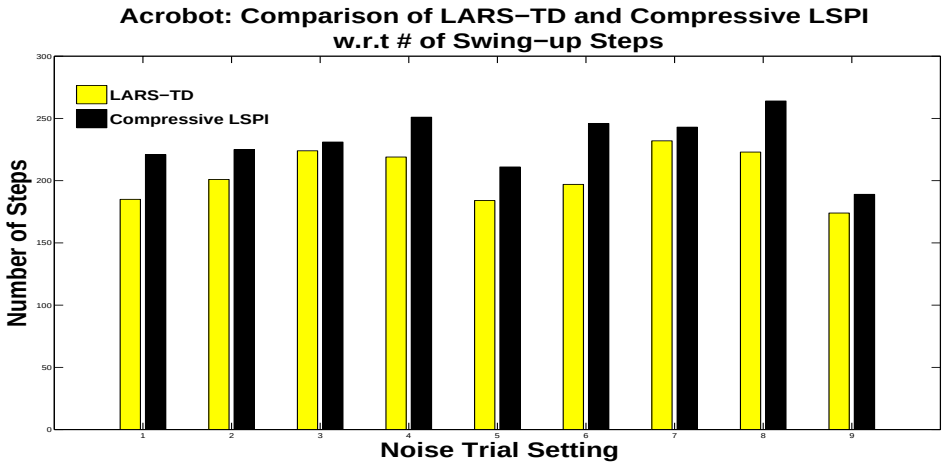

Figure 5: Acrobot: Comparison of Compressive Reinforcement Learning and LARS-TD in Noisy Setting

References

[1] Avrim Blum. Random Projection, Margins, Kernels, and Feature-Selection. *LNCS*, 3940:52–68, 2005.

[2] Yaakov Engel. Gaussian process reinforcement learning. In *Encyclopedia of Machine Learning*, pages 439–447. 2010.

[3] Mohammad Ghavamzadeh, Alessandro Lazaric, Odalric-Ambrym Maillard, and Remi Munos. LSTD with Random Projections. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2010.

[4] R. Hecht-Nielsen. *Computational Intelligence: Imitating Life*, chapter Context vectors: general purpose approximate meaning representations self-organized from raw data, pages 43–56. IEEE Press, 1994.

[5] Jeff Johns, Marek Petrik, and Sridhar Mahadevan. Hybrid least-squares algorithms for approximate policy evaluation. *Machine Learning*, 76:243–256, 2009.

[6] 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.

[7] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[8] Alessandro Lazaric, Mohammad Ghavamzadeh, and Remi Munos. Finite-Sample Analysis of LSTD. In *Proceedings of 27 th International Conference on Machine Learning*, pages 615–622, 2010.

[9] Sridhar Mahadevan and Mauro 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.

[10] Odalric-Ambrym Maillard and Remi Munos. Compressed least-squares regression. In *Proceedings of Advances in Neural Information Processing Systems 22*, pages 1213–1221, 2009.

[11] M. L. Puterman. *Markov Decision Processes*. Wiley Interscience, New York, USA, 1994.

[12] Bruno Scherrer. Should one compute the temporal difference fix point or minimize the bellman residual? the unified oblique projection view. In *Proceedings of 27 th International Conference on Machine Learning*, pages 52–68, 2010.

[13] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch II. *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

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

[15] Gavin Taylor and Ronald Parr. Kernelized value function approximation for reinforcement learning. In *Proceedings of 27 th International Conference on Machine Learning*, pages 1017–1024, 2009.

[16] Yaakov Tsaig and David L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.

[17] Huizhen Yu and Dimitri P. Bertsekas. New error bounds for approximations from projected linear equations. Technical report, Dept. Computer Science, Univ. of Helsinki, 2008.