Dominant Invariant Subspace for Feature Construction in Reinforcement Learning

Tao Huang
Student ID: 2018533172
Email: huangtao1@shanghaitech.edu.cn

Yintian Zhang
Student ID: 2020232168
Email: zhangyt@shanghaitech.edu.cn

Abstract

Generally accompanied with large-scale state spaces, reinforcement learning problems under various backgrounds are naturally addressed through the idea of value function approximation. Linear value function approximation (LVFA), one of the most common and essential approaches, has relatively higher sample efficiency to achieve stable value function and provides advanced insight to nowadays network-based methods. The main challenge of LVFA lies in how to construct reasonable features since prior knowledge about optimal value function is rarely available in practice. Based on the subspace invariance, we propose a new method for feature construction through orthogonal iteration with strongly theoretical guarantees. Moreover, we conduct numerical simulations to verify the empirical outperformance of our method over various baseline algorithms.

I. Introduction

Recent years have witnessed sensational progress of reinforcement learning (RL) in many challenging sequential decision-making problems, such as playing the game of Go [1] [2], robotic control [3] and autonomous driving [4]. The key idea under RL is modeling the agent to perform sequential decision-making by interacting with environment formulated as a Markov decision process (MDP).

In general, RL settings are commonly accompanied with large-scale state spaces, making how to ensure scalability a challenging open problem. Many existing methods based on value function approximation have been proposed to express the value function with some appropriate features [5]–[7]. Linear value function approximation (LVFA), representing the value function as a linear combination of given features, serves as a common and essential approximation approach among these methods. Specifically, it has relatively higher sample efficiency to estimate the value function stably due to the simple way of encoding prior knowledge. Moreover, many existing deep RL methods [1]–[4] commonly set a linear output layer where LVFA can be suitably applied to explore insights and interpretability of such network-based methods.

The key problem in LVFA is how to generate reasonable features that are sufficiently representative to approximate the true value function. This is regraded as a hard task since prior knowledge about true value function is rarely available in practice. Considerable effort has been dedicated to address such a problem in roughly two perspectives. One called feature selection attempts to select reasonable ones from a set of presented candidate features [8]–[13]. The other known as feature construction, which is the exact focus of this paper, aims to construct representative features automatically without given sets of features [14]–[18].

In this work, we propose a method that selects features from the transition matrix and the reward function. Specifically, it adopts orthogonal iteration to find orthonormal features and utilizes subspace invariance to reach great empirical performance with theoretical guarantees. This method is also extended to the case with large-scale state spaces by means of generating features directly from raw input data.

The rest of this paper is organized as follows. In Section II, we describe the fundamental framework and the notations related to LVFA. Then we give a brief overview and criticisms of the existing work in Section III. Our contribution including algorithm design and performance analysis is presented in Section IV, followed by our simulation results and analysis in Section V. Finally, Section VI concludes this paper with a few remarks on future work.

II. Framework and Notation

In this section, we give a brief overview of the fundamental background on linear value function approximation and feature selection in reinforcement learning. Mathematical notations, meanwhile, are introduced for clearer expression throughout this paper.

A. Markov Decision Process

We consider a sequential decision problem formulated as a Markov decision process (MDP) under reinforcement learning settings. Such a process is usually defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the transition probabilities from current state $s \in \mathcal{S}$ to any underlying states $s' \in \mathcal{S}$ given action $a \in \mathcal{A}$; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the reward function whose element $\mathcal{R}(s, a)$ is the expected instantaneous reward collected by the agent in state s by taking action a; $\gamma \in [0,1]$ is the discount factor that determines the present value of future reward.

Based on the aforementioned definitions, we define policy $\pi : \mathcal{S} \to \mathcal{A}$ as the behavior of the agent in MDPs. Specifically, at each time slot t, $a_t \sim \pi(\cdot|s_t)$ denotes a mapping from state s_t to the distribution over the whole action space $\mathcal{A}$. It basically characterizes the agent's way of decision-making under given state $s \in \mathcal{S}$. Accordingly, we define value function $v_\pi(s)$ as the expected discounted cumulative reward (return) starting from state $s \in \mathcal{S}$, which can be expressed as

$$v_\pi(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \mathcal{R}(s_t, a_t) \middle| a_t \sim \pi(\cdot|s_t), s_0 = s\right]. \tag{1}$$

By denoting $\boldsymbol{v}^\pi$ as the state-values in a vector form under a given policy π, we have the following Bellman equation [19]:

$$\boldsymbol{v}^\pi = \boldsymbol{r}^\pi + \gamma P^\pi \boldsymbol{v}^\pi, \tag{2}$$

where P^π and $\boldsymbol{r}^\pi$ are the transition probability in a matrix form and rewards in a vector form under policy π, respectively.

B. Linear Value Function Approximation

Linear value function approximation plays a key role in solving MDPs, especially with large-scale state spaces $\mathcal{S}$. It characterizes the value function in a parametric way, approximating the value of states as the linear combination of a set of k linear independent features $\Phi = \{\phi_1, ..., \phi_k\}$:

$$\hat{\boldsymbol{v}}^\pi = \Phi \boldsymbol{w}, \tag{3}$$

where $\phi_i \in \mathbb{R}^{|\mathcal{S}|}$ denotes the feature vector over state space $\mathcal{S}$ and $\boldsymbol{w} \in \mathbb{R}^k$ is the weight vector of the features. The task is how to find an appropriate weight vector $\boldsymbol{w}$ to approximate the true value function accurately.

1) Fixed-Point Method: Methods in [8]–[10] have been proposed to estimate a reasonable weight vector $\boldsymbol{w}$ given the features Φ. It is further proved in [20] that there exists unique weight vector $\boldsymbol{w}_\Phi^\pi$ which can be found by linear *fixed-point method*:

$$\hat{\boldsymbol{v}}^\pi = \Phi \boldsymbol{w}_\Phi^\pi = \Pi_\sigma(\boldsymbol{r}^\pi + \gamma P^\pi \Phi \boldsymbol{w}_\Phi^\pi). \tag{4}$$

where Π_σ represents the the σ-weighted L_2 projection into $span(\Phi)$ and σ is a state weighting distribution, which is reasonably assumed unweighted in this paper due to the independence with projection weights. It can be solved by least square approach or by applying orthogonal projector to both sides of (2). Both of them give the $\boldsymbol{w}_\Phi^\pi$ as:

$$\begin{aligned} \boldsymbol{w}_\Phi^\pi &= \Phi^\dagger(\boldsymbol{r}^\pi + \gamma P^\pi \Phi \boldsymbol{w}_\Phi^\pi) \\ &= (I - \gamma \Phi^\dagger P^\pi \Phi)^{-1} \Phi^\dagger \boldsymbol{r}^\pi, \end{aligned} \tag{5}$$

where $\Phi^\dagger = (\Phi^T\Phi)^{-1}\Phi^T$ is the Moore-Penrose pseudo-inverse of the features Φ [21]. The results have been extended to Q-functions in [17] with theoretical guarantees.

2) Linear Model: Another approach of desired unique weight vector $\boldsymbol{w}$ is by constructing a linear model in which the transition matrix P^π and rewards $\boldsymbol{r}^\pi$ are compressed as $P_\Phi^\pi \in \mathbb{R}^{k\times k}$ and $\boldsymbol{r}_\Phi^\pi \in \mathbb{R}^k$:

$$P_\Phi^\pi = \Phi^\dagger P^\pi \Phi, \quad \boldsymbol{r}_\Phi^\pi = \Phi^\dagger \boldsymbol{r}^\pi. \tag{6}$$

The above expressions follow from solving the least-square problem with the goal aiming to predict the next expected feature values and rewards, respectively. It has been proved that the solution of such a model is equivalent to which in (5) [14]. Specifically, the solution yields by solving the "compressed" Bellman equation:

$$\begin{aligned} \boldsymbol{w}_\Phi^\pi &= \boldsymbol{r}_\Phi^\pi + \gamma P_\Phi^\pi \boldsymbol{w}_\Phi^\pi \\ &= (I - \gamma P_\Phi^\pi)^{-1} \boldsymbol{r}_\Phi^\pi, \end{aligned} \tag{7}$$

which can be simplified as the form in (5) by substituting for P_Φ^π and $\boldsymbol{r}_\Phi^\pi$.

3) Bellman Operator and Bellman Error: To simplify the notion for operating value function, we hereby define the Bellman operator T^π on the value function $\boldsymbol{v}$ as

$$T^\pi \boldsymbol{v} = \boldsymbol{r}^\pi + \gamma P^\pi \boldsymbol{v}. \tag{8}$$

$\boldsymbol{v}^\pi$ is known as a fixed point of T^π, *i.e.* $T^\pi \boldsymbol{v}^\pi = \boldsymbol{v}^\pi$. Note that defining $\boldsymbol{v}^0 = \boldsymbol{r}^{\boldsymbol{\pi}}$, then $\boldsymbol{v}^{t+1} = (T^\pi)^t \boldsymbol{r}^\pi$, results in the value iteration algorithm where $\boldsymbol{v}^t \to \boldsymbol{v}^\pi$ as t increases. We will see that this property is naturally adopted to construct features in III-A. The Bellman error/residual is introduced to analyze the error under the value function $\boldsymbol{v}^\pi$ given the transition matrix P^π and rewards $\boldsymbol{r}^\pi$ [22], which is defined as

$$BE(\boldsymbol{v}) = T^\pi \boldsymbol{v} - \boldsymbol{v}. \tag{9}$$

When the Bellman error is 0, the value function $\boldsymbol{v}$ is at the fixed point. Otherwise, the distance between $\boldsymbol{v}$ and $\boldsymbol{v}^\pi$ is bounded in terms of the Bellman error:

$$\|\boldsymbol{v} - \boldsymbol{v}^\pi\|_\infty \leq \|\boldsymbol{v} - T^\pi \boldsymbol{v}\|_\infty + \|T^\pi \boldsymbol{v} - \boldsymbol{v}^\pi\|_\infty \leq BE(\boldsymbol{v}) + \gamma \|\boldsymbol{v} - \boldsymbol{v}^\pi\|_\infty \leq \frac{BE(\boldsymbol{v})}{1-\gamma}. \tag{10}$$

where $\|\boldsymbol{v}\|$ refers to the ℓ_∞ norm of a vector $\boldsymbol{v}$.

In the context of linear value function approximation, such a metric is naturally extended to quantify the quality of the selected features. Specifically, the Bellman error is redefined as the error in the linear fixed-point value function $\boldsymbol{v}_\Phi^\pi$ for given features Φ:

$$\begin{aligned} BE(\Phi) &= T^\pi(\Phi \boldsymbol{w}_\Phi^\pi) - \Phi \boldsymbol{w}_\Phi^\pi \\ &= \boldsymbol{r}^\pi + \gamma P^\pi \Phi \boldsymbol{w}_\Phi^\pi - \Phi \boldsymbol{w}_\Phi^\pi. \end{aligned} \tag{11}$$

The following theorem specifies that Bellman error can be decomposed into two separate sources of error.

Theorem 1. *For any MDP with given feature Φ and policy π, the Bellman error of a value function $\hat{\boldsymbol{v}}^\pi = \Phi \boldsymbol{w}_\Phi^\pi$ can be decomposed as [14]:*

$$\begin{aligned} BE(\Phi) &= \underbrace{(\boldsymbol{r}^\pi - \Phi \boldsymbol{r}_\Phi^\pi)}_{\Delta_R^\pi} + \gamma \underbrace{(P^\pi \Phi - \Phi P_\Phi^\pi)}_{\Delta_\Phi^\pi} \boldsymbol{w}_\Phi^\pi \\ &\triangleq \Delta_R^\pi + \gamma \Delta_\Phi^\pi \boldsymbol{w}_\Phi^\pi, \end{aligned} \tag{12}$$

where Δ_R^π and Δ_Φ^π corresponds to reward error and per-feature error, respectively.

In the next sections, we will show that such a decomposition can guide the process of generating features.

III. Related Work

There has been great interest in recent years in feature selection and automating feature construction for reinforcement learning. The main difference between these two approaches is that feature selection assumes a reasonable set of candidate features are presented to the learner, while such a presumably given set is rarely available in feature-construction's view.

Accordingly, for the former, the task is to select a small set of features to prevent over-fitting and expensive computational cost. This is difficult and requires a deep understanding of the domain. For the latter, the focus is how to automatically construct features representative enough to approximate the real value function. Meanwhile, priority should be given to scalability to ensure practical feasibility. In the remainder of this section, we give a brief overview of the existing work in the perspective of the two aforementioned approaches, followed by the criticisms of these work.

A. Overview of the Existing Work

1) Feature Selection: For the task of linear value function approximation, Least-Squares Temporal Difference (LSTD) algorithms [8]–[10] provide a basic approach for learning the value function using only trajectories generated by the system. However, these methods are prone to over-fitting and are computationally expensive, especially when the number of features is large compared to the number of training samples.

To overcome such a challenge, Kolter et al. [11] proposed a regularization framework as LARS-TD for linear value function approximation within LSTD. It employs the technique of regularization to find the sparse solution and therefore serves as an effective method for feature selection:

$$\begin{aligned} \boldsymbol{w}_\Phi^\pi &= (\Phi^T \Phi + \beta I)^{-1} \Phi^T (\boldsymbol{r}^\pi + \gamma P^\pi \Phi \boldsymbol{w}_\Phi^\pi) \\ &= \left(\Phi^T (\Phi - \gamma P^\pi \Phi) + \beta I\right)^{-1} \Phi^\pi \boldsymbol{r}^\pi, \end{aligned}$$

where $\beta \in [0, \infty)$ is a l_1 regularization parameter.

Johns et al. [12] formulated the L_1 regularized linear fixed point problem (Subsection II-B) as a linear complementarity problem (LCP):

$$\begin{gathered} A^{-1} = \Phi^T (\Phi - \gamma P^\pi \Phi), \quad b = \Phi^T \boldsymbol{r}^\pi, \quad c = b - A\boldsymbol{w} \\ \boldsymbol{w} = A^{-1} b + A^{-1}(-c). \end{gathered}$$

It allows the appropriate utilization of the rich theory of LCPs and provides strong theoretical guarantees with fast performance. Besides, policy iteration method is found more effective through this approach than previous methods.

Petrik et al. [13] considered feature selection as an optimization problem with particular L_1 regularization (RALP). Specifically, they integrated the approximate linear programming (ALP) with L_1 regularization to automate feature selection and alleviate the need for all constraints in standard ALP:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{\rho}^T \Phi \boldsymbol{w} \\ s.t. \quad & T^\pi(\Phi \boldsymbol{w}) \leq \Phi \boldsymbol{w} \quad \|\boldsymbol{w}\|_1 \leq \psi, \end{aligned}$$

where $\boldsymbol{\rho}$ is a distribution over the initial states (*i.e.,* $\sum_{s \in \mathcal{S}} \rho(s) = 1$), and $\|\boldsymbol{w}\|_1 \leq \psi$ is L_1 constraint. Their method offers some strong guarantees, but is not well-suited to noisy sampled data.

2) Feature Construction: To understand the relationship between the error in the linear model and the Bellman error, Parr et al. [14] proved that Bellman error can be decomposed into two separate source of error (Theorem 1),

$$BE(\Phi) = \underbrace{(\boldsymbol{r}^\pi - \Phi \boldsymbol{r}^\pi_\Phi)}_{\Delta^\pi_R} + \gamma \underbrace{(P^\pi \Phi - \Phi P^\pi_\Phi)}_{\Delta^\pi_\Phi} \boldsymbol{w}^\pi_\Phi$$

Recall that Δ^π_R and Δ^π_Φ corresponds to *reward error* and *per-feature error*, respectively. It provides a theoretical framework for constructing the basis feature functions, giving insights into the behavior of existing feature-construction algorithms.

Petrik et al. [15] proposed a method aiming to achieve no reward error. They build a Krylov basis [21] in terms of powers of transition matrix P multiplied by reward $\boldsymbol{r}$ as the features:

$$\Phi^K_k = Krylov_k(\boldsymbol{r}) = \{P^{i-1}\boldsymbol{r} : 1 \leq i \leq k\}.$$

As the true value function is conceptually defined as $\boldsymbol{v}^\pi = \sum_{t=1}^\infty \gamma^t (P^\pi)^t \boldsymbol{r}^{\boldsymbol{\pi}}$, their method is regarded as an intuitive design for value iteration, where the value functions of all iterations can be exactly represented by such a Krylov basis Φ^K_k.

Parr et al. [16] modeled the problem of constructing features as basis expansion. That is, given a set of basis functions (features) $\Phi^B_k = \{\phi_1, ..., \phi_k\}$ and the a linear fixed point solution, what is a good ϕ_{k+1} to extend the basis? They addressed this sequential problem by generating the next feature as the Bellman error $BE(\Phi^B_k)$ of the current fixed point solution $\boldsymbol{v}^\pi_{\Phi_k}$,

$$\phi_{k+1} = T^\pi \boldsymbol{v}^\pi_{\Phi_k} - \boldsymbol{v}^\pi_{\Phi_k} = BE(\Phi_k), \quad \Phi_{k+1} = \{\Phi^B_k, \phi_{k+1}\},$$

where ϕ_{k+1} is called as Bellman error basis functions. Parr et al. [14] have shown that Krylov bases and BEBF span the same approximation space $span\{\Phi^K_k\} = span\{\Phi^B_k\}$.

Song et al. [17] developed a theory of linear value function encoding. Inspired by the decomposition of Bellman error (Theorem 1), they aim to encode a reduced features that are representative enough to approximate the next expected raw features $P^\pi A$ and next reward $\boldsymbol{r}^\pi$. In order to determine the reasonable structure of encoder and decoder, they define *predictively optimal set* of features as follows:

$$AE_\pi D_\pi = [P^\pi A, \boldsymbol{r}^\pi]$$

where D_π and E_π are encoder and decoder, respectively. They have proved that the Bellman error is zero if such a tuple $\langle A, E_\pi, D_\pi \rangle$ can be found. Therefore, the proposed algorithm linear feature discovery (LFD) attempts to minimize the distance from $AE_\pi D_\pi$ to $[P^\pi A, \boldsymbol{r}^\pi]$ based on gradient descent method.

Behzadian et al. [18] utilized the low-rank factorization to construct features directly from raw features, specifically applying singular value decomposition to the compressed transition matrix P^π and including reward as one of the features as well. Their method (SVD+R) is proved as efficient method in both small-scale case and large-scale case with bounded theoretical Bellman error. The main difference between SVD+R and LFD is that the former attempts to reduce the raw features through encoder, while the latter dedicates to predict the next expected raw features through low-dimension representations directly.

B. Criticism of the Existing Work

LSTD-based algorithms [8]–[10] built a basic approach for approximation value function under LVFA's settings. But these methods do not consider the scale of features, leading to the failure of handling the over-fitting and expensively computational cost. To deal with such a challenge, Kolter et al. [11] proposed a regularization framework as LARS-TD for LVFA within LSTD. However, this method is not well-integrated with least squares policy iteration (LSPI) [23] algorithm due to the hardly any exploitation of previous fixed point solution. Based on the LCP, Johns et al. [12] proposed an algorithm that is well-employed by policy iteration. Nonetheless, the incorporation between policy improvement and LCP is considerably inflexible so that practical feasibility comes as a new challenge. Petrik et al. [13] considered feature selection from the view of optimization with particular L_1 regularization. But their analysis does not address the conditions that would guarantee sparse RALP solutions and it is not well-suited to noisy, sampled data.

Recall that feature selection assumes that reasonable features are available to learners. Therefore, in practice, these methods [8]–[13] do not address the problem of how to generate representative features in the first place. This naturally leads to the researches of feature construction which is considered a more basic problem.

Parr et al. [14] proved that Bellman error can be decomposed into two separate sources of error. But their theoretical results are not extended to the controlled case. Specifically, they did not provide the connection between model error given a set of features and Bellman error of a Q-function based on these features. Petrik et al. [15] proposed a method BEBF of great intuition, aiming to eliminate the reward error. However, the transition matrix must be approximated in a tabular manner and the resulting basis Φ_k^K is not orthonormal, incurring the high computational expense. Parr et al. [16] proposed using Bellman error to guide the sequential construction of features. But this method is not scalable since both large-scale state spaces and large number of features contribute to the time consumption. Song et al. [17] extended the connection between model error and Bellman error to the controlled case and developed linear feature encoding theory for LVFA. However, their method exists two main limits. On one hand, the proposed predictively optimal tuple $\langle A, E_\pi, D_\pi \rangle$ (III-A2) is almost impossible to be found in practice. On the other, the utilization of iterative learning strategy introduces an additional Bellman error, while theoretical analysis are not clear in this case. Behzadian et al. [18] applied singular value decomposition to approximate the transition matrix P^π. Nonetheless, the assumption about the low-rank transition matrix is deemed unreasonable, since such low-rank matrices are rare in practice. Therefore, the bound of Bellman error is theoretically loose and empirically large, leading to the mediocre performance in experiments.

IV. New contribution

Our method is highly related to the algorithm in [17] [18]. The impovements mainly lies in the point that, with milder assumption and great scalability, our algorithm theoretically achieves a tighter Bellman error bound and empirically reaches better performance.

In the remainder of this section, we give specific illustration of the proposed method DISCO for constructing features from transition matrix and rewards. First, we describe the method assuming that it is possible to represent the value function in a tabular form. Subsequently, we present a extended version of this method to fit the infinite state spaces to ensure scalability, treating raw feature inputs (such as images) as intermediate simplifying features.

Algorithm 1: Orthogonal Iteration

```
Input: $\mathbf{A} \in \mathbb{C}^{n \times n}$ and desired number of output vectors $k$.
Initialization: $j = 0$, semi-unitary matrix $\mathbf{V}^{(0)} \in \mathbb{C}^{n \times k}$;
repeat
    $\tilde{\mathbf{V}}^{(j+1)} = \mathbf{A}\mathbf{V}^{(j)}$;
    $\mathbf{V}^{(j+1)}\mathbf{R}^{(j+1)} = \tilde{\mathbf{V}}^{(j+1)}$;        /* normalization; perform thin QR for $\tilde{\mathbf{V}}^{(j+1)}$ */
    $\{\lambda_1^{(j+1)}, \lambda_2^{(j+1)}, \dots, \lambda_r^{(j+1)}\} = \sigma\left((\mathbf{V}^{(j+1)})^H \mathbf{A}\mathbf{V}^{(j+1)}\right)$;
    $j := j + 1$;
until a stopping rule is satisfied;
Output: $\mathbf{V}^{(j)}$.
```

A. DISCO for Finite State Space

Our algorithm design is essentially guided by the Theorem 1, where the key idea is to minimize the per-feature error Δ_Φ^π and reward error Δ_R^π separately. For the former, we eliminate such an error $\|P^\pi\Phi - \Phi P_\Phi^\pi\|_2$ by utilizing subspace invariance defined as $P^\pi\Phi \in span\{\Phi\}$. In particular, we apply orthogonal iteration to construct orthonormal features for the dominant invariant subspace of transition matrix P^π. For the latter, we achieve zero error by the means of including rewards $\boldsymbol{r}^\pi$ in the features. The intuition comes directly from the properties of orthogonal projection that $\boldsymbol{r}^\pi$ itself is in the range space of Φ.

Algorithm 2: DISCO: Dominant Invariant Subspace for Feature Construction

```
Input: Transition matrix $P^\pi$, rewards $\boldsymbol{r}^\pi$, and number of features $k$.
Compute orthogonal Iteration: $\mathbf{V} = Algorithm\ 1\,(P^\pi, k)$;
Approximate features: $\Phi = [\mathbf{V}, \boldsymbol{r}^\pi]$;
Output: $\Phi$.
```

Note that the existence of eigendecomposition is not a necessary condition here, since we only require a few basis vectors of dominant invariant subspace as features to approximate the value function. This attractive property ensures both low computational expense and practical feasibility. The following theorem specifies that the separate design of our algorithm implies a zero Bellman error.

Theorem 2. *Given featrues Φ output from Algorithm 2, the Bellman error $BE(\Phi) = 0$ under policy π.*

Proof. Substitute Φ into the definition of compressed transition matrix P_Φ^π (6):

$$\begin{aligned} P_\Phi^\pi &= \Phi^\dagger P^\pi \Phi \\ &= (\Phi^T \Phi)^{-1} \Phi^T P^\pi \Phi \\ &= \Phi^T P^\pi \Phi \\ &= \Phi^T \Phi \Lambda \\ &= \Lambda, \end{aligned}$$

where the third and the fifth equality comes from the fact that the output basis are orthonormal (ensured by normalized QR), and the forth equality follows from the subspace invariance $P^\pi \Phi = \Phi \Lambda$. Substituting it into the expression of per-feature error Δ_Φ^π (Theorem 1), we get:

$$\begin{aligned} \|\Delta_\Phi^\pi\|_2 &= \|P^\pi \Phi - \Phi P_\Phi^\pi\|_2 \\ &= \|\Phi\Lambda - \Phi\Lambda\|_2 \\ &= 0, \end{aligned}$$

where the second equality follows from the subspace invariance. Then we derive the bound of reward error:

$$\begin{aligned} \|\Delta_R^\Phi\|_2 &= \|\boldsymbol{r}^\pi - \Phi\Phi^\dagger \boldsymbol{r}^\pi\|_2 \\ &= \|\boldsymbol{r}^\pi - \Phi(\Phi^T\Phi)^{-1}\Phi^T \boldsymbol{r}^\pi\|_2 \\ &= \|\boldsymbol{r}^\pi - \Phi\Phi^T \boldsymbol{r}^\pi\|_2 \\ &= 0, \end{aligned}$$

where the last equality follows directly from the properties of orthogonal projection since $\boldsymbol{r}^\pi$ itself is in the range space of Φ. By applying triangular inequality and Cauchy Schwarz's inequality, we directly derive the Bellman error bound:

$$\begin{aligned} \|BE(\Phi)\|_2 &= \|\Delta_R^\pi + \gamma \Delta_\Phi^\pi \boldsymbol{w}_\Phi^\pi\|_2 \\ &\leq \|\Delta_R^\pi\|_2 + \gamma \|\Delta_\Phi^\pi\|_2 \|\boldsymbol{w}_\Phi^\pi\|_2 \\ &= 0, \end{aligned}$$

which implies Bellman error $BE(\Phi) = 0$ due to the non-negativity of norm. □

For most RL problem, the scale of the state space is almost infinite so that the transition matrix and reward vector used in Algorithm 2 will be too large to work with. To make our method more practical, we present an extended version of DISCO in the next subsection.

B. DISCO for Infinite State Space

Inspired by the approaches proposed by [17] [18], we first use a source of raw features, such as the image in video games, to compress the matrix of transition probability matrix. Then we conduct Algorithm 2 on these compressed transition matrix and reward vector, followed by the multiplication with raw feature, to reduce the dimension of representations. We give mathematical descriptions in the following part for clearer expression.

Let A be an $|S| \times l$ matrix, where each row corresponds to a state and each column corresponds to one raw feature, such as the grey value of a pixel. The compressed transition matrix P_A^π and the compressed rewards $\boldsymbol{r}_A$ are defined as

$$P_A^\pi = A^\dagger P^\pi A \triangleq A^\dagger A', \qquad \boldsymbol{r}_A^\pi = A^\dagger \boldsymbol{r}^\pi,$$

where A' denotes the expected next raw features. The dimensions of P_A^π and $\boldsymbol{r}_A^\pi$ are now corresponding to the number of raw features. Once they are estimated, we conduct DISCO algorithm 2 with them instead of P^π and $\boldsymbol{r}^\pi$ to get the encoded features Φ_E, which is of a dimension $l \times k$. Finally, we construct the ultimate features Φ_D by multiplying with A. We show the pseudo code of such process in Algorithm 3.

Algorithm 3: DISCO for Infinite State Space

```
Input: Raw features $A$, expected next raw features $A'$, rewards $\boldsymbol{r}^\pi$, and number of features $k$.
Compute compressed transition matrix: $P_A^\pi = A^\dagger A'$;
Compute compressed reward: $\boldsymbol{r}_A^\pi = A^\dagger \boldsymbol{r}^\pi$;
Encoded features: $\Phi_E = Algorithm\ 2\,(P_A^\pi, \boldsymbol{r}_A^\pi, k)$;        /* Encode raw features $A$ into $\Phi_E$ */
Decoded features: $\Phi_D = A\Phi_E$;        /* Decode $\Phi_E$ into a reduced size features $\Phi_D$ */
Output: $\Phi_D$.
```

This algorithm allows the subsequent operations on the encoded features Φ_E rather than raw features to lower the computational expense. Simple and practical as it is, we still catch that the final features Φ_D is a linear combination of the raw features, implying they cannot express more complex value functions. This is intuitively reasonable in the view of bias-variance tradeoff, where additional error is introduced—akin to bias—but sampling error is reduced—akin to variance. The following theorem gives the bound of new-introduced error and how it affect the Bellman error with respect to features Φ_D.

Theorem 3. *Assume that the raw features A and encoded features Φ_E are normalized, i.e., $\|A\|_2 = \|\Phi_D\|_2 = 1$. Then:*

$$\|\Delta_P^{\Phi_D}\|_2 \leq \|\Delta_P^A\|_2 + \|\Lambda - P_{\Phi_D}\|_2, \quad \|\Delta_R^{\Phi_D}\|_2 \leq \|\Delta_R^A\|_2 + \|\Delta_{R_A}^{\Phi_E}\|_2$$

where the superscript and subscript of Δ represents the features and the model (i.e., $\Delta_P^A = PA - AP_A$), respectively. Here we omit the the policy π in superscripts for simplicity.

Proof. By adding a zero item $(AP_A\Phi_E - AP_A\Phi_E)$ and applying triangular inequality and Cauchy Schwarz's inequality, we get:

$$\begin{aligned}
\|\Delta_P^{\Phi_D}\|_2 &= \|P\Phi_D - \Phi_D P_{\Phi_D}\|_2 \\
&= \|P\Phi_D - AP_A\Phi_E + AP_A\Phi_E - \Phi_D P_{\Phi_D}\|_2 \\
&\leq \|PA\Phi_E - AP_A\Phi_E\|_2 + \|AP_A\Phi_E - A\Phi_E P_{\Phi_D}\|_2 \\
&\leq \|PA - AP_A\|_2 + \|\Phi_E\Lambda - \Phi_E P_{\Phi_D}\|_2\|A\|_2 \\
&\leq \|\Delta_P^A\|_2 + \|\Lambda - P_{\Phi_D}\|_2
\end{aligned}$$

Similarly, we derive the bound of reward error as:

$$\begin{aligned}
\|\Delta_R^{\Phi_D}\|_2 &= \|\boldsymbol{r} - A\boldsymbol{r}_A + A\boldsymbol{r}_A - \Phi_D\boldsymbol{r}_{\Phi_D}\|_2 \\
&\leq \|\boldsymbol{r} - A\boldsymbol{r}_A\|_2 + \|A\boldsymbol{r}_A - A\Phi_E\boldsymbol{r}_{\Phi_D}\|_2 \\
&\leq \|\boldsymbol{r} - A\boldsymbol{r}_A\|_2 + \|\boldsymbol{r}_A - \Phi_E\boldsymbol{r}_{\Phi_D}\|_2 \\
&= \|\Delta_R^A\|_2 + \|\Delta_{R_A}^{\Phi_E}\|_2
\end{aligned}$$

Note that the normalization of raw features A in Theorem 3 has no effect on approximating the value function, since the the linear fixed point is invariant with such operation. However, it does change the scale of weight vector $\boldsymbol{w}_\Phi$. □

C. Relationship to LFD and SVD+R

Both DISCO and LFD in [17] apply the idea of reducing the representations' dimension for greater scalability. But the main difference between these two algorithms is that the former attempts to reduce the raw features through an encoder, while the latter dedicates to predict the next expected raw features through low-dimension representations directly. We consider ours a more practical method due to two aspects. On one hand, the proposed predictively optimal tuple $\langle A, E_\pi, D_\pi\rangle$ (III-A2) is almost impossible to be found in practice. On the other, the utilization of iterative learning strategy introduces an additional Bellman error and time consumption, and theoretical analysis are not clear in this case.

As for DISCO and SVD+R proposed in [18], they both construct ultimate features by the multiplication with raw features. Nonetheless, the assumption about the low-rank transition matrix in SVD+R is deemed unreasonable, since such low-rank matrices are rare in practice. Rather, this assumption is not necessary in DISCO. The bound of Bellman error in SVD+R, therefore, is theoretically loose and empirically large, leading to the mediocre performance in experiments comparing to DISCO.

V. Numerical results

In this section, we empirically compare some methods that we have discussed in previous sections. Here we use an image-based version of the Cart-Pole benchmark to compare the solution quality of DISCO, SVD+R and LFD algorithms. In the following subsections, we first introduce the basic settings of Cart-Pole. Simulation results are presented next, followed by our evaluation and analysis.

A. Cart-Pole

The Cart-Pole environment is a complex reinforcement learning benchmark problem. The controller should push the cart left or right to ensure the pole on the top of the cart stay vertical as long as possible. The measurement of the performance of the controller is the balanced steps in each episode. And in our experiment, the controller must learn a policy by merely observing the *image* of the Cart-Pole game screen without direct access to the position of the cart or the angle of the pole.

To obtain training data, we collected the specified number of trajectories with the starting angle and angular velocity sampled uniformly on $[-0.1, 0.1]$. The cart position and velocity are set to zero at each episode. The algorithm was given three consecutive, rendered, gray-scale images of the Cart-Pole. Each image has 39×50 pixels, so the raw state is a $39\times50\times3 = 5850$-dimensional vector. We chose three frames to preserve the Markov property of states without manipulating the Cart-Pole simulator in OpenAI Gym. We used $k = 50$ features for all three methods in the experiments, which is similar to state properties in [17].

The training data sets are produced by running the Cart-Pole game with random policy for $[100, 200, 300, 400, 500, 600]$ episodes. And we run the policy iteration algorithm until there is no change in $A' = P^{\pi}A$. The learned policy was later evaluated 100 times to obtain the average number of balancing steps.

B. Simulation Results and Analysis

Figure 1 displays the average number of balanced steps of three algorithms using the same training data sets. This result shows that the DISCO algorithm outperforms SVD+R and LFD in most cases. In particular, DISCO and SVD+R achieve nearly same performance when the the scale of sampled state space is relatively small. But it is remarkably superior than others under the large-scale cases. Such results verify the theoretical conclusion we have proved in Section IV (such as scalability and error bound), and further uncover the advantages over the baseline algorithms.

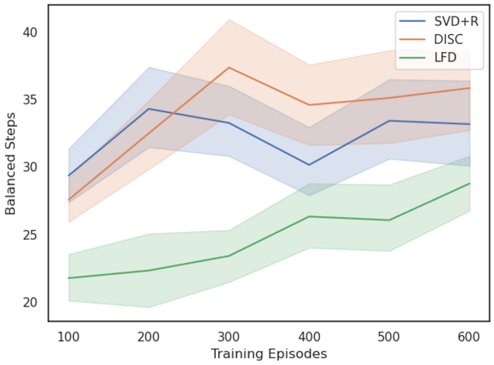

Fig. 1. Average number of balancing steps with $k = 50$.

VI. Conclusions and Future work

In this paper, we studied the problem of feature construction for linear value function approximation. Utilizing subspace invariance, we proposed DISCO algorithm based on orthogonal iteration method that ensures a theoretically tighter bound of Bellman error. Through the numerical results, we showed that our algorithm also outperforms other baseline methods empirically.

Although the linear value function approximation is less competitive than modern deep reinforcement learning, linear feature construction is still essential for nowadays network-based methods. Our future work aims to conjugate LVFA with current DRL methods, thus providing insights into the reinforcement learning problems.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *ArXiv*, vol. abs/1312.5602, 2013.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[3] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2016.

[4] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *ArXiv*, vol. abs/1610.03295, 2016.

[5] R. Sutton and A. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 16, pp. 285–286, 2005.

[6] C. Szepesvari, "Algorithms for reinforcement learning," in *Algorithms for Reinforcement Learning*, 2010.

[7] R. Wang, S. Du, L. F. Yang, and R. Salakhutdinov, "On reward-free reinforcement learning with linear function approximation," *ArXiv*, vol. abs/2006.11274, 2020.

[8] S. J. Bradtke and A. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, pp. 33–57, 2004.

[9] R. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 2005.

[10] A. Geramifard, M. Bowling, and R. Sutton, "Incremental least-squares temporal difference learning," in *AAAI*, 2006.

[11] J. Z. Kolter and A. Ng, "Regularization and feature selection in least-squares temporal difference learning," in *ICML '09*, 2009.

[12] J. Johns, C. Painter-Wakefield, and R. Parr, "Linear complementarity for regularized policy evaluation and improvement," in *NIPS*, 2010.

[13] M. Petrik, G. Taylor, R. Parr, and S. Zilberstein, "Feature selection using regularization in approximate linear programs for markov decision processes," in *ICML*, 2010.

[14] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. Littman, "An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning," in *ICML '08*, 2008.

[15] M. Petrik, “An analysis of laplacian methods for value function approximation in mdps,” in *IJCAI*, 2007.

[16] R. Parr, C. Painter-Wakefield, L. Li, and M. Littman, “Analyzing feature generation for value-function approximation,” in *ICML ’07*, 2007.

[17] Z. Song, R. Parr, X. Liao, and L. Carin, “Linear feature encoding for reinforcement learning,” in *NIPS*, 2016.

[18] B. Behzadian, “Feature selection by singular value decomposition for reinforcement learning,” 2019.

[19] M. Puterman, “Markov decision processes: Discrete stochastic dynamic programming,” in *Wiley Series in Probability and Statistics*, 1994.

[20] D. Koller and R. Parr, “Computing factored value functions for policies in structured mdps,” in *IJCAI*, 1999.

[21] G. G. H. Et.Al, “Matrix computations, 4th edition,” 2016.

[22] R. J. Williams and L. Baird, “Tight performance bounds on greedy policies based on imperfect value functions,” 1993.

[23] M. Lagoudakis and R. Parr, “Least-squares policy iteration,” *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, 2003.