Low-rank Feature Selection for Reinforcement Learning

Bahram Behzadian and **Marek Petrik**
Department of Computer Science
University of New Hampshire
{bahram, mpetrik}@cs.unh.edu

Abstract

Linear value function approximation is a common approach to solving large reinforcement learning problems. Because designing good approximation features is difficult, automatic feature selection is an important research topic. We propose a new method for feature selection, which is based on a low-rank factorization of the transition matrix. Our approach derives features directly from high-dimensional raw inputs, such as image data. The method is easy to implement using SVD, and our experiments show that it is faster and more stable than alternative methods.

Introduction

Most reinforcement learning methods for solving problems with large state spaces rely on some form of value function approximation (Sutton and Barto 1998; Szepesvári 2010). Linear value function approximation is one of the most common and simplest approximation methods, expressing the value function as a linear combination of features, which are provided in advance.

While linear value function approximation is less powerful than modern deep reinforcement learning, it is still important in many domains. In particular, linear models are interpretable and need relatively few samples to reliably compute a value function. Features also make it possible to encode prior knowledge conveniently. Finally, the last layer in deep neural networks, used for reinforcement learning, often calculates a linear combination of the underlying neural network features.

A significant limitation of linear value function approximation is that it requires good features, which is, features that can approximate the optimal value function well. This is often difficult to achieve since good a priori estimates of the optimal value functions are rarely available. Considerable effort has therefore been dedicated to methods that can automatically construct useful features for linear value function approximation. Examples of methods that construct features, or select good features from a large set, include proto-value functions (Mahadevan and Maggioni 2007), diffusion wavelets (Mahadevan and Maggioni 2006), Krylov bases (Petrik 2007), BEBF (Parr et al. 2008), L_1-regularized TD (Kolter and Ng 2009), and ALP (Petrik et al. 2010).

Recently, Song et al. (2016) proposed a new feature selection method, linear feature discovery (LFD) which can reliably map many raw low-quality features—such a sequence of images—to a few meaningful ones. This method constructs a linear *encoder* and a linear *decoder* that preserve the Markov property, that is, the encoder maps the raw features of each state to a low-dimensional space in such a way that the transition to the next state is linear. The decoder then maps states back to the high-dimensional space. Although the efficiency of this method has been shown empirically, a theoretical explanation of when and how it works has been lacking.

As with all data-driven methods, feature construction (or selection) must make some simplifying assumptions about the problem structure, which reduces the number of necessary samples. We show, in this paper, that LFD assumes that the matrix of transition probabilities can be approximated using a *low-rank* matrix. Using low-rank approximation has led to significant successes in several machine learning domains, including collaborative filtering (Murphy 2012), and more recently Markov chains (Rendle, Freudenthaler, and Schmidt-Thieme 2010) and reinforcement learning (Ong 2015).

As another contribution, we propose Low-Rank Approximation (LRA), a new feature selection method based on the low-rank factorization of the transitions and rewards. Our approach simplifies LFD, speeds it up by orders of magnitude, and is easier to implement and analyze. Just like LFD, it leverages the low rank of the transition probability matrix to dramatically reduce the number of samples necessary to reliably approximate the value function. The advantage of using truncated SVD for Low-rank approximation over coordinate decent (used by LFD) is that SVD has a closed form solution and it is proven to be optimal under certain conditions. On the other hand, the coordinate descent approach does not guarantee to provide an optimal solution. For example, when the objective function is a non-smooth multivariate function, coordinate decent most likely finds a local minimum. LRA also provides an upper bound for Bellman error function.

The remainder of the paper is organized as follows. First, we describe the general framework of linear value function approximation and some new properties that are relevant to feature selection. Next, we show how LFD assumes that

the transition matrix is low-rank and how to speed up the method. Then, we describe LRA, the new feature construction method. We also present connections with other feature construction algorithms and empirically compare LRA with LFD.

Linear Value Function Approximation

In this section, we summarize the relevant background on linear value function approximation and feature construction. We also derive a new representation for computing fixed point solutions, which will be instrumental in understanding LFD and deriving LRA.

We consider a reinforcement learning problem formulated as a Markov decision processes (MDP) with states $\mathcal{S}$, actions $\mathcal{A}$, transition probabilities $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and rewards $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (Puterman 2005). The values $P(s, a, s')$ denote the probability of transitioning to state s' after taking action a in state s. Our objective is to compute a stationary policy π that maximizes the expected $\gamma-$discounted infinite-horizon return. It is well-known that the value function v^π for a policy π must satisfy the following equality (e.g., (Puterman 2005)):

$$\boldsymbol{v}^\pi = \boldsymbol{r}^\pi + \gamma P^\pi \boldsymbol{v}^\pi \ , \tag{1}$$

where P^π and $\boldsymbol{r}^\pi$ are the matrix of transition probabilities and the vector of rewards, respectively, for the policy π.

Value function approximation becomes necessary in MDPs with large state spaces. Linear value function approximation approximates the value function as a linear combination of features $\phi_1, \ldots, \phi_k \in \mathbb{R}^{|\mathcal{S}|}$, which are real vectors over states. Using a vector representation, the approximate value function $\tilde{\boldsymbol{v}}^\pi$ can be expressed as follows:

$$\tilde{\boldsymbol{v}}^\pi = \Phi \, \boldsymbol{w},$$

for some vector $\boldsymbol{w} = \{w_i, \ldots, w_k\}$ of scalar weights that quantify the importance of features. Here, Φ is the feature matrix of dimensions $|\mathcal{S}| \times k$; the columns of this matrix are the features ϕ_i.

Numerous algorithms for computing linear value approximation have been proposed (see e.g., (Sutton and Barto 1998; Lagoudakis and Parr 2003; Szepesvári 2010)). In this work, we focus on fixed-point methods that compute the *unique* vector of weights $\boldsymbol{w}^\pi_\Phi$ that satisfy the projected version of equality (1):

$$\begin{aligned} \boldsymbol{w}^\pi_\Phi &= (\Phi^\top \Phi)^{-1} \Phi^\top (\boldsymbol{r}^\pi + \gamma P^\pi \Phi \boldsymbol{w}^\pi_\Phi) \\ &= \Phi^+ (\boldsymbol{r}^\pi + \gamma P^\pi \Phi \boldsymbol{w}^\pi_\Phi) \ . \end{aligned} \tag{2}$$

Here, the operator X^+ denotes the Moore-Penrose pseudo-inverse of X (e.g., Golub and Van Loan (2013)). This equation can be derived by applying the orthogonal projection operator $\Phi(\Phi^\top \Phi)^{-1} \Phi^\top$ to both sides of (1).

The fixed-point solution in (2) can also be seen as a *linear compression* of the transition matrix and reward vector (Parr et al. 2008; Szepesvári 2010). The "compressed" transition matrix P^π_Φ and reward vector $\boldsymbol{r}^\pi_\Phi$ are computed as:

$$P^\pi_\Phi = (\Phi^\top \Phi)^{-1} \Phi^\top P^\pi \Phi \qquad \boldsymbol{r}^\pi_\Phi = (\Phi^\top \Phi)^{-1} \Phi^\top \boldsymbol{r}^\pi \tag{3}$$

The fixed-point weights $\boldsymbol{w}^\pi_\Phi$ are then computed as a value function for this compressed model to satisfy the following set of linear equations:

$$\boldsymbol{w}^\pi_\Phi = \boldsymbol{r}^\pi_\Phi + \gamma P^\pi_\Phi \boldsymbol{w}^\pi_\Phi \ . \tag{4}$$

The following proposition shows that the compressed model is a solution to an optimization problem. This property, which we believe has not been pointed out before, will serve to provide the new interpretation of LFD (Song et al. 2016).

Proposition 1. *The compressed transition probability matrix P^π_Φ and rewards $\boldsymbol{r}^\pi_\Phi$ in* (3) *are the optimal solutions to the following problems:*

$$\begin{aligned} P^\pi_\Phi &= \arg\min_W \|\Phi W - P^\pi \Phi\|^2_F \\ \boldsymbol{r}^\pi_\Phi &= \arg\min_w \|\Phi \boldsymbol{w} - \boldsymbol{r}^\pi\|^2_2 \end{aligned}$$

Before proving the proposition, it is important to describe its intuitive meaning. The interpretation for $\boldsymbol{r}^\pi_\Phi$ is straightforward as it simply minimizes the distance between the true and approximate rewards. To understand the objective for P^π_Φ, recall that the operator Φ lifts a compressed value function $\boldsymbol{w}$ to a value function $\boldsymbol{v}$ in the original state space. Thus, the operator $P^\pi \Phi$ applies the original transition probabilities to a lifted value function, while the operator ΦW first applies the compressed transition probabilities and then lifts the result to the full original state space. The compressed matrix P^π_Φ is chosen to minimize the difference between these two operators. Also, Theorem 2 below shows that the objectives in Proposition 1 above can be used directly to bound the Bellman error.

Proof of Proposition 1. The result follows by algebraic manipulation from Equation (4) in (Parr et al. 2008), but we provide a direct and simpler proof.

The expression for $\boldsymbol{r}^\pi_\Phi$ is simply the solution to the orthogonal projection problem. The expression for P^π_Φ follows from the optimal solution to the Frobenius linear regression. To derive the optimal solution, recall that $\|A\|^2_F = \mathrm{Tr}(A^\top A)$ and therefore:

$$\|\Phi W - P^\pi \Phi\|^2_F = \mathrm{Tr}(\Phi W - P^\pi \Phi)^\top (\Phi W - P^\pi \Phi)$$

Taking the derivative of this function with respect to W and noting that $\frac{d}{dW} \mathrm{Tr}\, f(W) = \mathrm{Tr}\, \frac{d}{dW} f(W)$, since trace is a linear operator, we get:

$$\begin{aligned} &\frac{d}{dW} \mathrm{Tr}(\Phi W - P^\pi \Phi)^\top (\Phi W - P^\pi \Phi) = \\ &\quad = W^\top \Phi^\top \Phi - \Phi^\top (P^\pi)^\top \Phi = \mathbf{0} \end{aligned}$$

Multiplying by $(\Phi^\top \Phi)^{-1}$ on the right finishes the proof. □

Since we want to construct features that can be used to represent a good value function, it is important to quantify how good such a function is. The standard bound on the performance loss of a policy computed using, for example, approximate policy iteration, can be bounded as a function of the Bellman error (e.g., (Williams and Baird 1993; Puterman 2005)). The following theorem shows that the

Bellman error can be decomposed into two components: an error in the compressed rewards and an error in the compressed transition probabilities.

Theorem 2 (Song et al. 2016)**.** *Given a policy π and features Φ, the Bellman error of a value function $v = \Phi \boldsymbol{w}_\Phi^\pi$ satisfies:*

$$\mathrm{BE}_\Phi = \underbrace{(\boldsymbol{r}^\pi - \Phi \boldsymbol{r}_\Phi^\pi)}_{\Delta_r^\pi} + \gamma \underbrace{(P^\pi \Phi - \Phi P_\Phi^\pi)}_{\Delta_P^\pi} \boldsymbol{w}_\Phi^\pi \, .$$

Notice that the terms Δ_r^π and Δ_P^π are related to the objective values in Proposition 1. The Bellman error can be upper bounded as follows:

$$\begin{aligned} \| \mathrm{BE}_\Phi \|_2 &\leq \|\Delta_r^\pi\|_2 + \|\Delta_P^\pi\|_2 \|\boldsymbol{w}_\Phi^\pi\|_2 \leq \\ &\leq \|\Delta_r^\pi\|_2 + \|\Delta_P^\pi\|_F \|\boldsymbol{w}_\Phi^\pi\|_2 \end{aligned}$$

The second inequality holds since $\|X\|_F \geq \|X\|_2$. Therefore, Proposition 1 shows that the compressed model is chosen to minimize an upper bound on the Bellman error.

Equipped with the properties above, we are ready to analyze when and why LFD works, as well as to propose a faster algorithm that uses only truncated SVD to compute the low-rank approximation.

LFD: Linear Feature Encoding

In this section, we describe a new interpretation of LFD as a low-rank approximation of the matrix of transition probabilities. Algorithm 1 illustrates a simplified version of the Linear Feature Discovery algorithm, which does not consider the rewards vector and approximates the value function instead of q-function. This helps to explain when one may expect the method to construct useful features and when it may fail. The original algorithm can be found in (Song et al. 2016).

Algorithm 1: LFD: Linear Feature Discovery for a fixed policy π (Song et al. 2016).

$D \leftarrow \mathrm{random}(k, m)$;
while *Not Converged* **do**
  // solve $\arg\min_E \|AED - P^\pi A\|_F^2$
  $E \leftarrow A^+ P^\pi A D^+$;
  // solve $\arg\min_D \|AED - P^\pi A\|_F^2$
  $D \leftarrow (AE)^+ P^\pi A$
end
return E

The LFD algorithm starts with a large number of raw features, such as ones that come from images. We denote the raw feature matrix as A. As with the feature matrix Φ, each column of A represents a raw feature, and each row accounts for a sampled state. The number of unique states sampled is n, and the number of raw features is m. Thus, the dimensions of A are $n \times m$.

It might be natural to ask why raw features must be compressed. Clearly, using Φ which is a *linear* transformation of the raw features in A will only further restrict the set of value functions that can possibly be represented. One could simply compute $\boldsymbol{w}_A^\pi$ using (2). If the number of samples is unlimited, then $\boldsymbol{w}_A^\pi$ is guaranteed to be no worse than $\boldsymbol{w}_\Phi^\pi$. Samples and computational power are usually limited, however, in which case $\boldsymbol{w}_A^\pi$ will almost surely overfit the sample. Restricting the set of features to Φ reduces the risk of overfitting and, indeed, serves as a regularization method.

LFD, a simplified version of which is summarized in Algorithm 1, selects k features that linearly combine the raw features in A using an *encoder* E^π with dimensions $m \times k$. A new encoder is constructed whenever approximating the value function of a policy π. Thus, the new feature matrix for a policy π is $\Phi^\pi = AE^\pi$. The counterpart to the encoder is a *decoder* D^π of dimensions $k \times m$. The decoder maps the smaller set of features to the raw features and only serves to evaluate the quality of the approximation. Song et al. (2016) introduce an encoder and decoder for both transition probabilities and rewards, but we focus only on the transition probabilities to simplify the presentation.

The LFD algorithm is motivated by the theory of *predictive optimal feature encoding* described in Song et al. (2016). An encoder E^π is *predictively optimal* if there exist decoders D_s^π and D_r^π such that:

$$\begin{aligned} AE^\pi D_s^\pi &= P^\pi A \\ AE^\pi D_r^\pi &= r^\pi \end{aligned}$$

The following theorem shows that a predictively optimal controller can be used to construct optimal features.

Theorem 3 (Theorem 7 in (Song et al. 2016))**.** *Let E^π be a* predictively optimal *controller. Then, the Bellman error with features $\Phi = AE^\pi$ is zero:* $\mathrm{BE}_\Phi = 0$.

In other words, Theorem 3 shows that if Φ can predict the next raw features $P^\pi A$ and rewards $\boldsymbol{r}^\pi$, then the approximate value function equals the true value function.

While the above-described motivation is compelling, it suffers from two main limitations. First, it does not allow for an approximately predictive optimality when $\|AE^\pi D_s^\pi - P^\pi A\|$ is merely small rather than being 0. Second, there is no guarantee that such an E^π exists for a given set of raw features A even if the number of features k is unlimited. To summarize, it is unlikely that a truly predictively optimal encoder exists and there are no approximation guarantees when it does not.

We argue that there is a better explanation for why LFD works and which assumptions it makes about the problem structure. It uses Proposition 1 to show that LFD approximates P_A^π using a low-rank matrix. Low-rank matrix factorization is a common approach to regularizing large and noisy matrices such as P_A^π. To show how LFD does this, consider the following low-rank version of the approximation in Proposition 1:

$$\min_W \|AW - P^\pi A\|_F^2 \quad \text{subject to} \quad \mathrm{rank}(W) \leq k \qquad (5)$$

Notice that without the rank constraint the solution would simply be $P_A^\pi = A^+ P^\pi A$.

Perhaps the most common and practical method for enforcing a rank constraint on a matrix, such as the W in (5),

is to express it as a product of two low-dimensional matrices. Thus let:

$$W = ED$$

for some E and D of dimensions $m \times k$ and $k \times m$. Replacing W by ED in (5) gives us a new optimization problem:

$$\min_{E \in \mathbb{R}^{m \times k}} \min_{D \in \mathbb{R}^{k \times m}} \|AED - P^\pi A\|_F^2 . \quad (6)$$

Notice that if there exists a *predictively optimal* encoder of dimension k then the optimal objective value in (6) is 0.

LFD, summarized in Algorithm 1, solves (6) by coordinate descent. It alternatively computes the optimal E and D values while assuming that the other one is fixed. Values E^π and D^π are then the minimizers of (6).

We believe that our explanation addresses the problems with the original motivation of LFD, which assumes there exist a perfect encoder. We show that it is not necessary to achieve the equality, but it is sufficient to achieve a small objective value in (6). Recall that the objective value in $\|AED - P^\pi A\|_F$ is directly related to the Bellman error, as Theorem 2 and the discussion below it show.

More important, our explanation shows that LFD is likely to work when the transition probability matrix P^π has a good low-rank approximation. This is particularly the case if the raw features are very expressive, such as when $A = \mathbf{I}$. When there is no such approximation, or if the raw features A are not good, then LFD is likely to lead to a significant Bellman error in approximating the value function and a potentially bad policy.

Previous work has established that LFD often constructs good features, but solving the optimization problem in (6) is challenging and can be time-consuming. In the next section, we propose a faster optimization method and empirically demonstrate its advantages.

LRA: Low-Rank Approximation for Feature Construction

In this section, we propose a new method for selecting features. It assumes that P^π is low rank, like LFD, but runs in orders of magnitude faster.

LRA is also based on low-rank approximation, but we solve a relaxed version the optimization problem in (6) as follows:

$$\min_{E \in \mathbb{R}^{m \times k}} \min_{D \in \mathbb{R}^{k \times m}} \|ED - A^+ P^\pi A\|_F^2 . \quad (7)$$

The optimization must also include the reward function and will therefore become:

$$\min_{E \in \mathbb{R}^{m \times k}} \min_{D \in \mathbb{R}^{k \times m+1}} \|AED - [P_A^\pi, \boldsymbol{r}_A^\pi]\|_F ,$$

since $P_A^\pi = A^+ P^\pi A$ and $\boldsymbol{r}_A^\pi = A^+ \boldsymbol{r}^\pi$.

The optimization problem (7) is closely related to the LFD optimization problem (6). The main difference between the two approaches is where the low-rank constraint is applied. LFD looks for a low-rank compressed model with a small number of features. LRA, on the other hand, first computes the transition probabilities matrix and afterward looks for a low-rank approximation. There is no significant difference between the Bellman error bounds that they imply. It is likely, however, that a more detailed theoretical analysis would reveal their differences, but this is beyond the scope of the present paper.

Although LRA's objective is quite similar to LFD's objective, it is nevertheless much easier to solve. As the *matrix approximation lemma*, or Eckart-Young-Mirsky theorem, shows, all that is needed to recover E and D is to compute the top k singular vectors of P_A^π (Eckart and Young 1936).

Assume the following singular value decomposition of P_A^π:

$$P_A^\pi = \sum_{i=1}^{m} \sigma_i u_i v_i^\top = U \Sigma V^\top,$$

where the singular values σ_i are sorted non-increasingly. We can define matrices $\Sigma_1 \in \mathbb{R}^{k \times k}$, $U_1 \in \mathbb{R}^{m \times k}$ and $V_1 \in \mathbb{R}^{m \times k}$ with the following decomposition:

$$U = [U_1 \quad U_2], \quad \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}, \quad V = [V_1 \quad V_2] .$$

The *matrix approximation lemma* then implies that an optimal (not unique) solution to (5) is:

$$E^\pi = U_1,$$
$$D^\pi = \Sigma_1 V_1^\top .$$

The matrices U_1, Σ_1, V_1 can be computed using the truncated SVD. Also, the optimal objective value is:

$$\left\| E^\pi D^\pi - A^+ P^\pi A \right\|_F = \sqrt{(\sigma_{k+1}^2 + \ldots + \sigma_m^2)}$$

We empirically analyze the performance of LRA over simple and moderately complex synthetic problems. We also discuss the advantages and disadvantages of LRA when compared to other feature construction methods.

Empirical Evaluation

Here, we compare our method (LRA) with linear feature discovery (LFD). We do not compare with other feature construction methods, since Song et al. (2016) present an extensive empirical comparison of LFD with radial basis functions (RBFs) (Lagoudakis and Parr 2003), and random projections (Ghavamzadeh et al. 2010), and others.

First, we compare the quality of LRA and LFD solutions on a range of synthetic randomly-generated problems. The goal is to ensure that the methods behave similarly regardless of the number of samples, or the type of the raw features that are used. Second, we use an image-based version of the cart-pole benchmark to show that the methods perform similarly even in more complicated settings. This more-complex problem is also used when comparing the computational complexity of the two approaches.

Synthetic Problems

To compare the low-rank approximation method (LRA) with the linear feature discovery algorithm (LFD), we start with an uncontrolled policy evaluation problem with a small

number of states. Since we assume a fixed policy throughout these experiments, we omit all references to it. The data matrix $A \in \mathbb{R}^{n \times m}$ only contains the states where n denotes the number of states and m the length of each *raw* feature, with $\Phi \in \mathbb{R}^{n \times k}$ using k features.

The synthetic problems that we use throughout this section have 100 states. The rewards $\boldsymbol{r} \in \mathbb{R}^{100}$ are generated uniformly randomly from the interval of $[0, 10)$. The transition probabilities $P \in [0, 1]^{100 \times 100}$ are generated from the uniform Dirichlet distribution. To ensure that the rank of P is at most 40, we compute P as a product $P = XY$, where X and Y are small-dimensional . The discount factor we use is $\gamma = 0.95$.

We now proceed by comparing LFD and LRA in a sequence of increasingly complex settings. In particular, we distinguish between cases based on whether the transition probabilities are sampled or known and whether the raw features are tabular or non-tabular. To evaluate the quality of the value function approximation, we compute the Bellman residual of the fixed-point value function, which is a standard metric used for this purpose. Recall that the Bellman error can be expressed as

$$\mathrm{BE} = \Delta_{\boldsymbol{r}} + \gamma \Delta_P \boldsymbol{w}_\Phi,$$

where $\boldsymbol{w}_\Phi$ is the value-function given in (4). All results we report in this section are an average of 100 repetitions of the experiments. All error plots show the L_2 norm of the Bellman error.

Case 1: Known transition probabilities and tabular raw features. In this case, the true transition probabilities P are assumed to be known and the raw features are an identity matrix: $A = \mathbf{I}$. The reward function is also known.

This is the simplest setting, under which LRA simply reduces to a direct low-rank approximation of the transition probabilities. That is, the LRA optimization problem reduces to:

$$\min_{E \in \mathbb{R}^{m \times k}} \min_{D \in \mathbb{R}^{k \times m}} \|ED - P\|_F^2 .$$

Similarly, the constructed features will be $\Phi = E$ and:

$$\Delta_P = PE - EP_\Phi \qquad P_\Phi = (E^\top E)^{-1} E^\top PE$$
$$\Delta_{\boldsymbol{r}} = \boldsymbol{r} - E\boldsymbol{r}_\Phi \qquad \boldsymbol{r}_\Phi = (E^\top E)^{-1} E^\top \boldsymbol{r}$$

Figure 1 depicts the Bellman error in this case. Note that the errors are 0 for $k \geq 40$. This is because the rank of P is 40. Since the transition probabilities are known precisely, both methods succeed in recovering them when $k \geq 40$. The results also show that the two methods perform identically in this simple setting. The only difference between them is the lower running time of LRA.

Case 2: Sampled transition probabilities and tabular raw features. The true transition probabilities are not known, and the transition matrix needs to be estimated from samples. The features are tabular and thus $A = \mathbf{I}$.

Instead of the transition probabilities matrix, we are given a set of simulated samples from the transition probabilities P and estimate $\tilde{A}$ and its corresponding output matrix $\tilde{A}' = PA$. We use 50 consecutive samples to estimate these values. In this case, LRA solves the following optimization problem:

$$\min_{E \in \mathbb{R}^{m \times k}} \min_{D \in \mathbb{R}^{k \times m}} \|ED - \tilde{A}^+ \tilde{A}'\|_F^2 .$$

The features that we use are $\Phi = \tilde{A}E$.

Figure 2 shows Bellman error for this case. For a small number of features, the methods are identical, but the solution quality of LFD degrades with respect to LRA for intermediate numbers of features. Since when $A = \mathbf{I}$ both LRA and LFD solve an identical optimization problem, we hypothesize that this drop in solution quality is due to the fact the LFD finds only locally optimal solutions. When matrix P is noisy, it appears that LFD fails to find the globally optimal solution.

Case 3: Known transition probabilities and image-based raw features. As in Case 1, we assume that the transition probabilities are known, but now the raw features A are not tabular, but rather simulate an image representation of states. The matrix A is generated by randomly allocated zeros and ones similar to the structure of a binary image.

Figure 3 compares the Bellman error between LFD and LRA features. Interestingly, just as in Case 1, the two methods have essentially the same performance. This is surprising because the corresponding optimization problems are quite different for structured raw features (that is when A is not an identity matrix).

Case 4: Sampled transition probabilities and image-based raw features. This case combines sampled transition probabilities with image-like features, as in Case 3.

Figure 4 compares the Bellman error between LFD and LRA features. Just as in Case 2, LRA appears to be much more stable when the transition probabilities are not known and must be sampled.

It is worth noting that this section deals with very small MDPs with only about 100 states. In actual MDP problems with enormous and high-dimensional state spaces, the gap between the Bellman error of the two methods would likely be larger. In the next section, we compare the two methods using a much larger and more challenging benchmark problem.

Cart Pole

These experiments evaluate the similarity between the linear feature encoding approach and our LRA method on a modified version of Cart-Pole, which is a more complex reinforcement learning benchmark problem. Unlike in the standard setting, the controller must learn a good policy by simply observing the *image* of the cart and the pole. The controller does not have direct access to the position of the cart or the angle of the pole. This problem is also large enough that the computational time plays a more important role, so we also compare the computational complexity of the two methods.

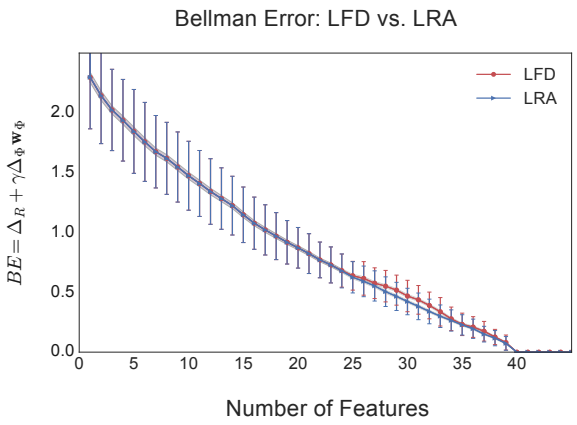

Figure 1: Known transition probabilities and tabular raw features (Case 1).

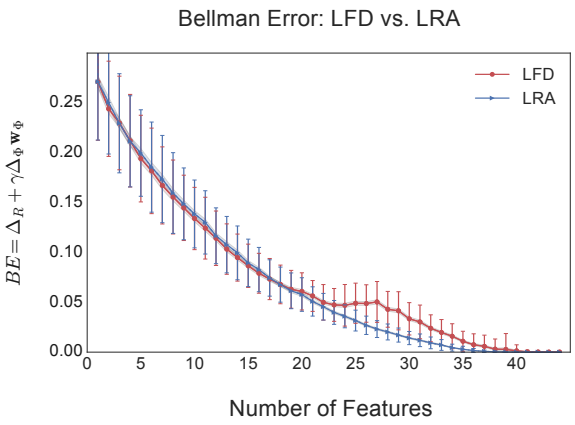

Figure 2: Sampled transition probabilities and tabular raw features (Case 2).

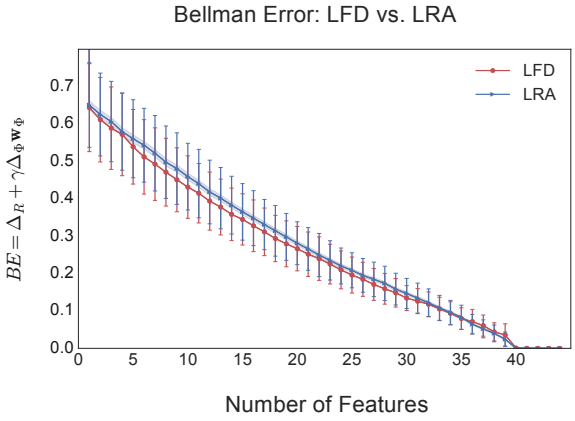

Figure 3: Known transition probabilities and image-based raw features (Case 3).

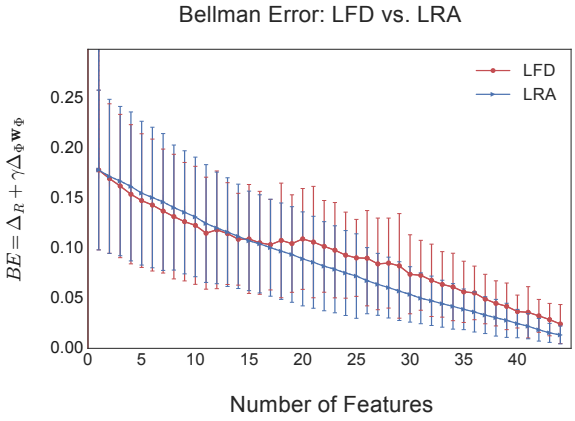

Figure 4: Known transition probabilities and image-based raw features (Case 4).

To obtain the training data, we collected the desired number of trajectories with the starting angle, and angular velocity sampled uniformly on $[0.1, 0.1]$. Cart position and velocity are set to zero at the beginning of each episode. None of the algorithms see the true state of cart pole. The algorithm was given two consecutive, rendered, gray-scale images of the cart pole. Each image has 52×35 pixels, so the raw state is a $52 \times 35 \times 2 = 3640-$dimensional vector. We used $k = 50$ features for both LFD and LRA similar to state properties in (Song et al. 2016).

We implemented LFD as described by Song et al. (2016) and followed an analogous setup when implementing LRA. The training data sets are produced by running the cart for $[200, 400, 600, 800]$ episodes with a random policy. We then run policy iteration to iterate up to 50 times or until there is no change in the $A' = PA$ matrix.

The learned policy was later assessed 50 times to obtain the average number of balancing steps. Figure 5 displays the average number of steps during which the pole kept its balance using the same training data sets for LFD and LRA. This result shows that the policies obtained from these methods are equivalent. The threshold for LFD was fixed to a small number 0.0003. Otherwise, LFD results in larger Bellman error in comparison to LRA in all synthetic problems.

We also evaluate the average running time of LFD and LRA for a single function call with $k = 50$. Figure 6 depicts the result of this comparison. The computation time of LRA grows very slowly as the number of training episodes increases; at 800 training episodes, the maximum number of episodes tested, LRA is 20 times faster that LFD. Therefore, LFD would likely be impractical in large problems with many training episodes.

Related Feature Selection Methods

Although numerous feature selection methods have been proposed in the last decade, we limit our discussion to those methods most relevant to LRA.

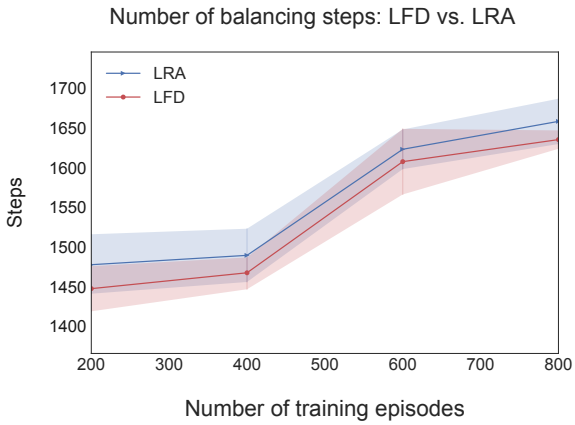

Figure 5: Number of balancing steps with $k = 50$

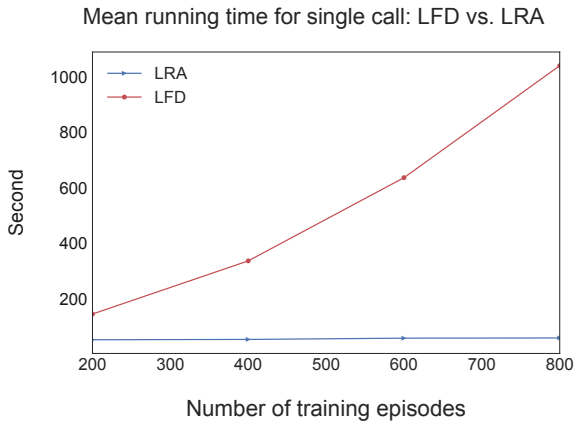

Figure 6: Mean running time with $k = 50$

Perhaps the simplest and most general method is based on random projections; it makes no assumptions, and it runs quickly (Ghavamzadeh et al. 2010) Another class of methods has been inspired by LASSO and uses L_1 regularization to select meaningful features (Kolter and Ng 2009; Petrik et al. 2010; Johns, Painter-Wakefield, and Parr 2010). These methods make no assumptions about the structure of the matrix of transition probabilities but simply seek to find a small set of relevant features. They are likely to work even when the transition matrix is not low-rank, but may require a larger set of samples.

A class set of methods that is most similar to LRA relies on the eigenvalues of some form of the probability matrix and includes proto-value functions and Krylov methods (Mahadevan 2005; Petrik 2007). The main difference is the objective and how the features are selected.

Probably the closest approach to LRA is a method for a low-rank approximation based on Robust PCA (Ong 2015); this method, however, has a much higher computational complexity and only works directly with the matrix of transition probabilities.

Conclusion

In this paper, we showed that LFD—a recently proposed feature selection method—is a form of low-rank approximation to the matrix of transition probabilities. When the transition matrix is low-rank or at least can be approximated well using a low-rank matrix, one would expect LFD to perform well. On the other hand, if no such approximation exists, the method is likely to fail.

The main limitation of LFD is that it involves solving a difficult optimization problem, which makes the algorithm impractical for solving problems with large state spaces. We address this limitation by proposing an alternative optimization algorithm, LRA, which involves a more direct low-rank matrix approximation. In our formulation, the low-rank approximation can be derived directly from an SVD of a "compressed" transition probabilities matrix.

Our empirical results show that LRA computes value functions that are as good as those that are computed using LFD but can do so several orders of magnitude faster. Even in the cart-pole problem, which is a relatively simple benchmark problem, we observed up to almost 20 fold speedup in feature computation time. Also, while LFD needs to know the number of features to select in advance, the SVD-based LRA can select the number of features on the fly based on the decay of singular values.

Acknowledgments

We thank the anonymous reviewers for detailed comments that helped to improve this paper significantly. This work was in part supported by the National Science Foundation under Grant No. IIS-1717368.

References

Eckart, G., and Young, G. 1936. The approximation of one-matrix by another of lower rank. *Psychometrika* 1:211218.

Ghavamzadeh, M.; Lazaric, A.; Maillard, O.-A.; and Munos, R. 2010. LSTD with Random Projections. In *Neural Information Processing Systems (NIPS)*.

Golub, G. H., and Van Loan, C. F. 2013. *Matrix Computations*.

Johns, J.; Painter-Wakefield, C.; and Parr, R. 2010. Linear Complementarity for Regularized Policy Evaluation and Improvement. *Advances in Neural Information Processing Systems (NIPS) 23* 1009–1017.

Kolter, J. Z., and Ng, A. Y. 2009. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th annual international conference on machine learning*, 521–528. ACM.

Lagoudakis, M. G., and Parr, R. 2003. Least-squares policy iteration. *Journal of machine learning research* 4(Dec):1107–1149.

Mahadevan, S., and Maggioni, M. 2006. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *Advances in neural information processing systems*, 843–850.

Mahadevan, S., and Maggioni, M. 2007. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research* 8(Oct):2169–2231.

Mahadevan, S. 2005. Proto-value functions: Developmental reinforcement learning. In *International Conference on Machine Learning*.

Murphy, K. 2012. *Machine Learning: A Probabilistic Perspective*.

Ong, H. Y. 2015. Value Function Approximation via Low Rank Models. *arXiv*.

Parr, R.; Li, L.; Taylor, G.; Painter-Wakefield, C.; and Littman, M. L. 2008. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, 752–759. ACM.

Petrik, M.; Taylor, G.; Parr, R.; and Zilberstein, S. 2010. Feature selection using regularization in approximate linear programs for Markov decision processes. In *International Conference on Machine Learning*.

Petrik, M. 2007. An analysis of laplacian methods for value function approximation in mdps. In *IJCAI*, 2574–2579.

Puterman, M. L. 2005. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *International conference on World wide web (WWW)*, 811–820.

Song, Z.; Parr, R. E.; Liao, X.; and Carin, L. 2016. Linear feature encoding for reinforcement learning. In *Advances in Neural Information Processing Systems*, 4224–4232.

Sutton, R. S., and Barto, A. 1998. *Reinforcement learning*.

Szepesvári, C. 2010. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning* 4(1):1–103.

Williams, R. J., and Baird, L. C. 1993. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, Technical report, College of Computer Science, Northeastern University.