Feature Selection by Singular Value Decomposition for Reinforcement Learning

Bahram Behzadian 1  Marek Petrik 1

Abstract

Linear value function approximation is a standard approach to solving reinforcement learning problems with a large state space. Automatic feature selection is an important research topic since designing good approximation features is difficult. 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.

1. Introduction

Most reinforcement learning methods for solving problems with large state spaces rely on some form of value function approximation (Sutton & 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 (Song et al., 2016). In our proposed method, we generate orthonormal basis functions from the transition matrix and the reward function that produce useful features directly from raw input data.

A significant limitation of linear value function approximation is that it requires good features, in other words, 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 or at least select them from a larger set. Examples of such methods include proto-value functions (Mahadevan & Maggioni, 2007), diffusion wavelets (Mahadevan & Maggioni, 2006), Krylov bases (Petrik, 2007), BEBF (Parr et al., 2007), L1-regularized TD (Kolter & Ng, 2009), and L1-regularized ALP (Petrik et al., 2010).

As with all data-driven methods, feature construction (or selection) must make some simplifying assumptions about the problem structure that reduce the number of samples needed. In this paper, we assume that the transition matrix that predicts expected next feature vector 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), reinforcement learning (Ong, 2015; Cheng et al., 2017), and more recently Markov chains (Rendle et al., 2010).

Our main contribution is a new feature selection method that uses singular value decomposition (SVD) to compute a low-rank factorization of the transitions matrix and approximate reward predictor given the raw input data. We also show that it is crucial to include the rewards as one of the features. In comparison to similar methods, our approach is more robust, faster by orders of magnitude on large training sets, and most importantly results in steadily lower Bellman error in linear fixed-point solution.

The remainder of the paper is organized as follows. In Section 2, we describe the general framework of linear value function approximation and properties that are relevant to feature selection. In Section 3, we describe the new feature construction method and analyze its approximation error. Then in Section 4, we present the connections with other feature construction algorithms, and, in Section 5, we empirically compare the most efficient methods.
2. 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 our method.

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

$$v^\pi = r^\pi + \gamma P^\pi v^\pi,$$

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

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 $\phi_1, \ldots, \phi_k \in \mathbb{R}^{|S|}$, which are real vectors over states. Using a vector representation, an approximate value function $\hat{v}^\pi$ can be expressed as: $\hat{v}^\pi = \Phi w$, for some vector $w = \{w_1, \ldots, w_k\}$ of scalar weights that quantify the importance of features. Here, $\Phi$ is the feature matrix of dimensions $|S| \times k$; the columns of this matrix are the features $\phi_i$.

Numerous algorithms for computing linear value approximation have been proposed (Sutton & Barto, 1998; Lagoudakis & Parr, 2003; Szepesvári, 2010). We focus on fixed-point methods that compute the unique vector of weights $w_\Phi$ that satisfy the projected Bellman equation (1):

$$w_\Phi = \Phi^+ (r^\pi + \gamma P^\pi \Phi w_\Phi),$$

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

The fixed-point solution to (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 $P_\Phi$ and reward vector $r_\Phi$ are:

$$P_\Phi = (\Phi^\top \Phi)^{-1} \Phi^\top P^\pi \Phi, \quad r_\Phi = (\Phi^\top \Phi)^{-1} \Phi^\top r^\pi.$$  

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

$$w_\Phi = r_\Phi + \gamma P_\Phi w_\Phi.$$  

Since we want to construct features that can be used to represent a good value function, it is essential to quantify the quality of such a function. 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 & Baird (1993)). The following theorem states that the Bellman error can be decomposed into two components: the error in the compressed rewards and in the compressed transition probabilities.

**Theorem 1** (Song et al. 2016). Given a policy $\pi$ and features $\Phi$, the Bellman error of a value function $v = \Phi w_\Phi$ satisfies:

$$\text{BE}_\Phi = \left(\frac{r^\pi - \Phi r_\Phi}{\Delta^\pi} + \gamma \frac{P^\pi \Phi - \Phi P_\Phi}{\Delta^\pi}\right) w_\Phi.$$

The Bellman error can be upper bounded as follows:

$$\|\text{BE}_\Phi\|_2 \leq \|\Delta^\pi\|_2 + \|\Delta^\pi\|_2 \|w_\Phi\|_2 \leq \|\Delta^\pi\|_2 + \|\Delta^\pi\|_2 \|w_\Phi\|_2$$

The second inequality holds since $\|X\|_F \geq \|X\|_2$.

3. SVD+R: A Low-rank Approximation for Feature Construction

In this section, we describe the proposed method for selecting features from singular value decomposition of transition probabilities and rewards. First, we describe the method assuming that it is possible to represent the value function in a tabular form. Since this approach does not scale, we describe an extension to infinite domains using raw feature inputs (such as images) as intermediate simplifying features, similarly to Linear Feature Discovery (LFD) in Song et al. (2016).

**Algorithm 1 SVD+R: Low-rank feature discovery**

**Input:** Transition matrix $P$, rewards $r$, and number of features $k + 1$

1. Compute SVD decomposition of $P$: $P = U \Sigma V^\top$
2. Assuming decreasing singular values in $\Sigma$, select the first $k$ columns of $U$: $U_1 \leftarrow [u_1, \ldots, u_k]$

**return:** Approximation features: $\Phi = [U_1, r]$.

An attractive property of our algorithm, summarized in Algorithm 1, is its simplicity and low computational complexity. Selecting the essential features only requires computing the singular value decomposition—for which many efficient methods exist—and augmenting the result with the
Feature Selection by Singular Value Decomposition for Reinforcement Learning

...
Note that the normalization of features required in Theorem 3 can be achieved by multiplying all features by an appropriate constant, and it does not affect the computed approximate value function in any way. It does, however, affect the scale of $w_\theta$.

4. Relationship to Linear Feature Discovery

The method that is closest to SVD+R is the Linear Feature Discovery (LFD) (Song et al., 2016). LFD is motivated by the theory of predictive optimal feature encoding. A low-rank encoder $E^\pi$ is predictively optimal if there exist decoders $D^s_\pi$ and $D^r_\pi$ such that:

$$A E^\pi D^s_\pi = P^\pi A, \quad AE^\pi D^r_\pi = r^\pi.$$

Song et al. (2016) show that when an encoder and decoder are predictively optimal, then the Bellman error is 0. Unfortunately, it is almost impossible to find problems in practice in which a predictively optimal controller exists. No bounds on the Bellman error are known when a controller is merely close to predictively optimal.

Both SVD+R and LFD construct a low-rank approximation of the transition matrix. Algorithm 2 in Song et al. (2016) shows the LFD algorithm. Recall that $A$ is the matrix of raw features, which are sampled from the state space, such as images of individual states. LFD corresponds to a coordinate descent that is often used for low-rank matrix completion e.g., (Hastie et al., 2015). SVD+R computes a similar low-rank decomposition using SVD. Using SVD is faster, offers error bounds, and performs as well or better in our empirical results.

5. Empirical Evaluation

In this section, we empirically compare the methods that we discussed in previous sections. Song et al. (2016) present an extensive empirical comparison of LFD with radial basis functions (RBFs) (Lagoudakis & Parr, 2003) and random projections (Ghavamzadeh et al., 2010). They show that LFD outperforms all of them. Here we focus on LFD, SVD+R, and RPr, which stands for random projection method. We use an image-based version of the Cart-Pole benchmark to compare the computational complexity and solution quality of SVD+R and LFD.

5.1. Cart-Pole

These experiments evaluate the similarity between the linear feature encoding approach and our SVD+R method on a modified version of Cart-Pole, which is a complex reinforcement learning benchmark problem. The controller must learn a good policy by merely observing the image of the Cart-Pole without direct access to the position of the cart or the angle of the pole. This problem is large enough that the computational time plays an important role, so we also compare the computational complexity of the two methods.

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 $39 \times 50$ pixels, so the raw state is a $39 \times 50 \times 3 = 5850$-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 both LFD and SVD+R 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 SVD+R. The training data sets are produced by running the cart for $[50, 100, 200, 400, 600]$ episodes with a random policy. We then run policy iteration to iterate up to 50 times or until there is no change in the $A' = P^\pi A$ matrix.

The learned policy was later evaluated 100 times to obtain the average number of balancing steps. Figure 1 displays the average number of steps during which the pole kept its balance using the same training data sets. For each episode, a maximum of 200 steps was allowed to run. This result shows that on the larger training sets the policies obtained from SVD+R and LFD are quite similar, but with small training sets SVD+R shows a better performance.

We also evaluated the average running time of LFD and SVD+R for obtaining the value function with $k = 50$. Figure 2 depicts the result of this comparison. The computation time of SVD+R grows very slowly as the number of training episodes increases; at 600 training episodes, the maximum number of episodes tested, SVD+R is 10 times faster than LFD. Therefore, LFD would likely be impractical in large problems with many training episodes.

6. Conclusion

We propose SVD+R, a new feature construction technique that computes a low-rank approximation of the transition probabilities. Our experimental results show that SVD+R works as well as LFD but with a much lower computational complexity. This addresses a central limitation of LFD and makes the algorithm more practical when solving large-scale problems. SVD+R also has better theoretical properties than LFD, and it is easier to implement and analyze.

It is unlikely that linear feature selection techniques can be competitive with modern deep reinforcement learning. Linear feature selection is nevertheless important. First,
it can be used in conjunction with deep learning methods where the raw features are sources from the neural net. Second, linear features can be used to gain additional insights into the given reinforcement learning problem. These are issues that we are planning to address in future work.

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


