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.

[1]Department of Computer Science, University of New Hampshire, Durham, NH, USA. Correspondence to: Bahram Behzadian <bahram@cs.unh.edu>, Marek Petrik <mpetrik@cs.unh.edu>.

Accepted at the FAIM workshop "Prediction and Generative Modeling in Reinforcement Learning", Stockholm, Sweden, 2018. Copyright 2018 by the author(s).

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), L_1-regularized TD (Kolter & Ng, 2009), and L_1-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 $\mathcal{S}$, 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 an action a in a 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 $\boldsymbol{v}^\pi$ for a policy π must satisfy the following Bellman equation (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, an approximate value function $\tilde{\boldsymbol{v}}^\pi$ can be expressed as: $\tilde{\boldsymbol{v}}^\pi = \Phi\, \boldsymbol{w}$, 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 (Sutton & Barto, 1998; Lagoudakis & Parr, 2003; Szepesvári, 2010). We focus on fixed-point methods that compute the *unique* vector of weights $\boldsymbol{w}_\Phi^\pi$ that satisfy the projected Bellman equation (1):

$$\boldsymbol{w}_\Phi^\pi = \Phi^+ (\boldsymbol{r}^\pi + \gamma P^\pi \Phi \boldsymbol{w}_\Phi^\pi) \ , \tag{2}$$

where Φ^+ denotes the Moore-Penrose pseudo-inverse of Φ (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_Φ^π and reward vector $\boldsymbol{r}_\Phi^\pi$ are:

$$P_\Phi^\pi = (\Phi^\top \Phi)^{-1} \Phi^\top P^\pi \Phi, \qquad \boldsymbol{r}_\Phi^\pi = (\Phi^\top \Phi)^{-1} \Phi^\top \boldsymbol{r}^\pi \ . \tag{3}$$

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

$$\boldsymbol{w}_\Phi^\pi = \boldsymbol{r}_\Phi^\pi + \gamma P_\Phi^\pi \boldsymbol{w}_\Phi^\pi \ . \tag{4}$$

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 π and features Φ, the Bellman error of a value function $v = \Phi \boldsymbol{w}_\Phi^\pi$ satisfies:*

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

The Bellman error can be upper bounded as follows:

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

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

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 reward function. As we show next, this simple approach is well-motivated by bounds on approximation errors and empirically achieves a smaller error than existing feature selection methods. It is important to note that including the vector of rewards $\boldsymbol{r}$ in the features is essential both to achieving error bounds and good solutions empirically.

One may expect that when the matrix P is of a rank at most k then using the first k singular vectors will result in no error. This is indeed the case. However, such low-rank matrices are rare in practice. Our next theorem shows that it is sufficient that the transition matrix P is close to a low-rank matrix for our method to achieve small approximation errors. In order to bound the error, let the SVD decomposition of P be as follows $\mathrm{SVD}(P) = U\Sigma V^\top$, where

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

The matrix U_1 has k columns so Algorithm 1 generates $\Phi = U_1$. The following theorem shows a bound on the error regarding the largest singular value for a vector not included in the features.

Theorem 2. *Given features Φ computed by Algorithm 1, the error terms in Theorem 1 can be bounded as $\|\Delta_P\|_2 \le \|\Sigma_2\|_2$ and $\|\Delta_r\|_2 = 0$.*

Proof of Theorem 2. From the definition of Δ_P and P_Φ we get the following equality:

$$\Delta_P = U\Sigma V^\top U_1 - U_1(U_1^\top U_1)^{-1} U_1^\top U \Sigma V^\top U_1$$

Recall that singular vectors are perpendicular which implies that $(U_1^\top U_1)^{-1} = \mathbf{I}$ and $U_1^\top U = \begin{bmatrix} \mathbf{I}_1 & 0 \end{bmatrix}$. Substituting these terms into the equality above, we get:

$$\begin{aligned} \|\Delta_P\|_2 &= \left\|(U\Sigma V^\top - U_1 \Sigma_1 V_1^\top) U_1\right\|_2 \\ &\le \left\|U\Sigma V^\top - U_1\Sigma_1 V_1^\top\right\|_2 \|U_1\|_2 \end{aligned}$$

Simple algebraic manipulation shows that $\left\|U\Sigma V^\top - U_1\Sigma_1 V_1^\top\right\|_2 = \|\Sigma_2\|_2$ and $\|U_1\|_2 \le 1$ because U is an orthogonal matrix. This establishes the inequality for Δ_P; the result for Δ_r follows directly from the properties of orthogonal projection since $\boldsymbol{r}$ itself is included in the features. □

Theorem 2 implies that if we choose Φ in a way that the singular values in Σ_2 are zero (when the transition matrix is low rank), Δ_P would be zero. That means that for a low-rank matrix, the error Δ_P will be zero and thus the approximation will be precise.

The transition matrix and reward vector used in Algorithm 1 will be too large to work with most problems of realistic size. To make our method practical, we take an approach proposed by Song et al. (2016). That is, we first use a source of raw features—such as the game image in video games, or the output from a deep learning classifier—to compress the matrix of transition probabilities. Then we run Algorithm 1 on this compressed transition matrix and reward vector.

Let A be an $|S| \times l$ matrix of l raw features. As with Φ, each row corresponds to one state, and each column corresponds to one *raw* feature, such as the gray color value of a pixel. The compressed transition matrix P_A and compressed rewards $\boldsymbol{r}_A$ are defined as in (3). The dimensions now correspond to the number of raw features, and these values can be easily estimated from samples. Once the values P_A and $\boldsymbol{r}_A$ are estimated, we can run Algorithm 1 with them instead of P and $\boldsymbol{r}$. To get the ultimate value of the features, it is then sufficient to compute $A\widehat{\Phi}$ where $\widehat{\Phi}$ is the output of Algorithm 1. The matrix $\widehat{\Phi}$ represents features for P_A and is of a dimension $l \times k$ where l is the number of raw features in A.

The approach of first using raw features and then computing a small number of useful features is simple and practical, but it is also essential to understand the consequences of relying on the raw features. Since our computed features are a linear combination of the raw features, they cannot express more complex value functions. Our feature reduction method thus introduces additional error—akin to bias—but reduces sampling error—akin to variance. The following theorem shows that the errors due to our approximation and using raw features merely add up with no additional interactions.

Theorem 3. *Assume that the raw features A for P and computed features $\widehat{\Phi}$ for P_A are normalized, such that $\|A\|_2 = \|\widehat{\Phi}\|_2 = 1$. Then:*

$$\|\Delta_P^{A\widehat{\Phi}}\|_2 \le \|\Delta_P^A\|_2 + \|\Delta_{P_A}^{\widehat{\Phi}}\|_2, \quad \|\Delta_r^{A\widehat{\Phi}}\|_2 \le \|\Delta_r^A\|_2 + \|\Delta_{r_A}^{\widehat{\Phi}}\|_2,$$

where the superscript of Δ indicates features for which the error is computed; for example $\Delta_{P_A}^{\widehat{\Phi}} = P_A\widehat{\Phi} - \widehat{\Phi} P_{A\widehat{\Phi}}$.

Proof. Due to lack of space, we show the result only for Δ_P; the result for Δ_r follows similarly. From the definition,

$$\left\|\Delta_P^{A\widehat{\Phi}}\right\|_2 = \left\|PA\widehat{\Phi} - A\widehat{\Phi} P_{A_{\widehat{\Phi}}}\right\|_2 .$$

Now by adding a zero ($AP_A\widehat{\Phi} - AP_A\widehat{\Phi}$) and applying triangle inequality, we get:

$$\begin{aligned} \left\|\Delta_P^{A\widehat{\Phi}}\right\|_2 &= \left\|PA\widehat{\Phi} - AP_A\widehat{\Phi} + AP_A\widehat{\Phi} - A\widehat{\Phi}P_{A_{\widehat{\Phi}}}\right\|_2 \le \\ &\le \left\|PA\widehat{\Phi} - AP_A\widehat{\Phi}\right\|_2 + \left\|AP_A\widehat{\Phi} - A\widehat{\Phi}P_{A_{\widehat{\Phi}}}\right\|_2 \le \\ &\le \|PA - AP_A\|_2 \left\|\widehat{\Phi}\right\|_2 + \left\|P_A\widehat{\Phi} - \widehat{\Phi}P_{A_{\widehat{\Phi}}}\right\|_2 \|A\|_2 . \end{aligned}$$

The theorem then follows directly from algebraic manipulation and the fact that the features are normalized. □

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_Φ.

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^π is *predictively optimal* if there exist decoders D_s^π and D_r^π such that:

$$AE^\pi D_s^\pi = P^\pi A \, , \qquad AE^\pi D_r^\pi = r^\pi \, .$$

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 39×50 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,

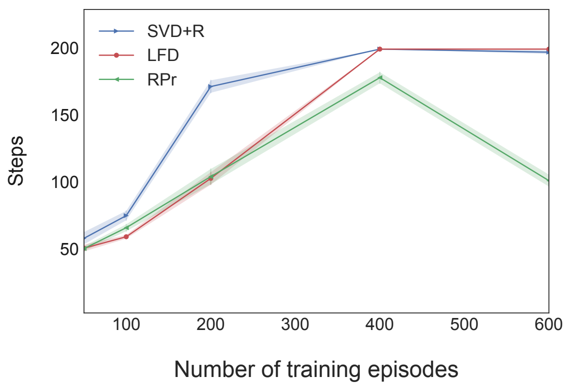

Figure 1. Average number of balancing steps with $k = 50$.

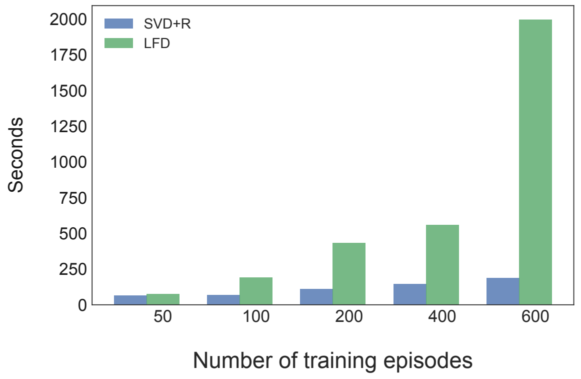

Figure 2. Mean running time for estimating the q-function with $k = 50$

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

Cheng, Bolong, Asamov, Tsvetan, and Powell, Warren B. Low-Rank Value Function Approximation for Co-optimization of Battery Storage. *IEEE Transactions on Smart Grid*, 3053, 2017.

Ghavamzadeh, Mohammad, Lazaric, Alessandro, Maillard, Odalric, and Munos, Rémi. LSTD with random projections. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 721–729, 2010.

Golub, Gene H and Van Loan, Charles F. *Matrix Computations*. 2013.

Hastie, Trevor, Mazumder, Rahul, Lee, Jason, and Zadeh, Reza. Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares. *Journal of Machine Learning Research*, 16:3367–3402, 2015.

Kolter, J Zico and Ng, Andrew Y. Regularization and feature selection in least-squares temporal difference learning. In *International Conference on Machine Learning (ICML)*, pp. 521–528. ACM, 2009.

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

Mahadevan, Sridhar and Maggioni, Mauro. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 843–850, 2006.

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

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

Ong, Hao Yi. Value Function Approximation via Low Rank Models. *arXiv*, 2015.

Parr, Ronald, Painter-Wakefield, Christopher, Li, Lihong, and Littman, Michael. Analyzing feature generation for value-function approximation. In *International Conference on Machine Learning (ICML)*, pp. 737–744, 2007.

Parr, Ronald, Li, Lihong, Taylor, Gavin, Painter-Wakefield, Christopher, and Littman, Michael L. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 752–759, 2008.

Petrik, Marek. An analysis of Laplacian methods for value function approximation in MDPs. In *International Joint Conference on Artificial Intelligence*, volume 35, pp. 2574–2579, 2007.

Petrik, Marek, Taylor, Gavin, Parr, Ron, and Zilberstein, Shlomo. Feature selection using regularization in approximate linear programs for Markov decision processes. In *International Conference on Machine Learning (ICML)*, 2010.

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

Rendle, Steffen, Freudenthaler, Christoph, and Schmidt-Thieme, Lars. Factorizing personalized Markov chains for next-basket recommendation. In *International Conference on World Wide Web (WWW)*, pp. 811–820, 2010.

Song, Zhao, Parr, Ronald E, Liao, Xuejun, and Carin, Lawrence. Linear feature encoding for reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 4224–4232, 2016.

Sutton, Richard S and Barto, Andrew. *Reinforcement learning*. 1998.

Szepesvári, Csaba. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, 2010.

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