An Analysis of Linear Models, Linear Value-Function Approximation, and Feature Selection for Reinforcement Learning

Ronald Parr[⋆] PARR@CS.DUKE.EDU
Lihong Li[†] LIHONG@CS.RUTGERS.EDU
Gavin Taylor[⋆] GVTAYLOR@CS.DUKE.EDU
Christopher Painter-Wakefield[⋆] PAINT007@CS.DUKE.EDU
Michael L. Littman[†] MLITTMAN@CS.RUTGERS.EDU

[⋆]Department of Computer Science, Duke University, Durham, NC 27708 USA

[†]Department of Computer Science, Rutgers University, Piscataway, NJ 08854 USA

Abstract

We show that linear value-function approximation is equivalent to a form of linear model approximation. We then derive a relationship between the model-approximation error and the Bellman error, and show how this relationship can guide feature selection for model improvement and/or value-function improvement. We also show how these results give insight into the behavior of existing feature-selection algorithms.

1. Introduction

Broadly speaking, there are two types of reinforcement-learning (RL) algorithms: model-free and model-based algorithms. Model-free approaches typically use samples to learn a value function, from which a policy is implicitly derived. In contrast, model-based approaches build a model of system behavior from samples, and the model is used to compute a value function or policy. Both approaches have advantages and disadvantages, and function approximation can be applied to either, to represent a value function or a model. Examples of function approximators include decision trees, neural networks, and linear functions.

The first contribution of this paper shows that, when linear value-function approximation is used for policy evaluation as in nominally model-free approaches such as linear TD learning (Sutton, 1988) or LSTD (Bradtke & Barto, 1996), the value function is *precisely* the same as the value function that results from an exact solution to a corresponding approximate, linear model, where the value function and linear model are defined over the same set of features.

This insight results in a novel view of the Bellman error and a deeper understanding of the problem of feature selection when linear function approximation is used. Specifically, we show that the Bellman error can be decomposed into two types of errors in the learned linear model: the reward error and the feature error. This decomposition gives insight into the behavior of existing approximation techniques, suggests new views of feature selection, and explains the behavior of existing feature-selection algorithms.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Formal Framework and Notation

We are interested in both controlled and uncontrolled Markov processes with a set S of states s and, when applicable, a set A of actions a. Our main results and experiments consider the uncontrolled or policy-evaluation case, but many of the ideas can be applied to the controlled case, as is discussed in more detail in Section 3.2.

We refer to the uncontrolled case as a Markov reward process (MRP): $M = (S, P, R, \gamma)$, and the controlled case as a Markov decision process (MDP): $M = (S, A, P, R, \gamma)$. Given a state s_i, the probability of a transition to a state s_j given action a is given by P^a_{ij} and results in an expected reward of R^a_i. In the uncontrolled case, we use P_{ij} and R_i to stand for the transitions and rewards.

We are concerned with finding value functions V that map each state s_i to the expected total γ-discounted reward for the process. In particular, we would like to find the solution to the Bellman equation

$$V[s_i] = \max_a (R^a_i + \gamma \sum_j P^a_{ij} V[s_j])$$

in the controlled case (the "max" and a's are eliminated from the equation in the uncontrolled case).

For any matrix, A, we use A^T to indicate the transpose of A and $span(A)$ to indicate the column space of A.

2.1. Linear Value Functions

In cases where the value function cannot be represented exactly, it is common to use some form of parametric value-function approximation, such as a linear combination of features or basis functions:

$$\hat{V} = \sum_{i=1}^{k} w_i \phi_i,$$

where $\Phi = \{\phi_1, \ldots, \phi_k\}$ is a set of linearly independent[1] basis functions of the state, with $\phi_i(s)$ defined as the value of feature i in state s. The vector $\mathbf{w} = \{w_1, \ldots, w_k\}$ is a set of scalar weights. We can think of Φ as a design matrix with $\Phi[i,j] = \phi_j(s_i)$, that is, the basis functions span the columns of Φ and the states span the rows. Expressing the weights $\mathbf{w}$ as a column vector, we have $\hat{V} = \Phi\mathbf{w}$.

Methods for finding a reasonable $\mathbf{w}$ given Φ and a set of samples include linear TD (Sutton, 1988), LSTD (Bradtke & Barto, 1996) and LSPE (Yu & Bertsekas, 2006). If the model can be expressed as a factored MDP, then the weights can be found directly (Koller & Parr, 1999). We refer to this family of methods as *linear fixed-point* methods because they all solve for the fixed point

$$\hat{V} = \Phi\mathbf{w}_\Phi = \Pi_\sigma(R + \gamma P \Phi \mathbf{w}_\Phi), \tag{1}$$

where Π_σ is an operator that is the σ-weighted L_2 projection into $span(\Phi)$, where σ is a state weighting distribution, typically the stationary distribution of P. If $\Sigma = diag(\sigma)$, $\Pi_\sigma = \Phi(\Phi^T\Sigma\Phi)^{-1}\Phi^T\Sigma$. In some cases, an unweighted projection (uniform σ) or some other σ is used. Most of our results do not depend upon the projection weights, so we shall assume uniform σ unless otherwise stated. Solving for $\mathbf{w}_\Phi$ yields:

$$\mathbf{w}_\Phi = (I - \gamma(\Phi^T\Phi)^{-1}\Phi^T P\Phi)^{-1}(\Phi^T\Phi)^{-1}\Phi^T R \tag{2}$$
$$= (\Phi^T\Phi - \gamma\Phi^T P\Phi)^{-1}\Phi^T R. \tag{3}$$

In this paper, we assume that P is known and that Φ can be constructed exactly, while in all but the factored model case, these would be sampled. This assumption allows us to characterize the representational power of the features as a separate issue from the variance introduced by sampling.

2.2. Linear Models

As in the case of linear value functions, we assume the existence of a set of linearly independent features $\phi_1 \ldots \phi_k$ for representing transition and reward models, with $\phi_i(s)$ defined as the value of feature i in state s. While value-function approximation typically uses features to predict values, we will consider the use of these features to predict *next* features. For feature vector $\Phi(s) = [\phi_1(s) \ldots \phi_k(s)]^T$, we define $\Phi(s'|s)$ as the random vector of next features:

$$\Phi(s'|s) \stackrel{s' \sim P(s'|s)}{=} [\phi_1(s'), \ldots, \phi_k(s')]^T,$$

our objective will be to produce a $k \times k$ matrix P_Φ that predicts expected next feature vectors,

$$P_\Phi{}^T \Phi(s) \approx E_{s' \sim P(s'|s)}\{\Phi(s'|s)\},$$

and minimizes the expected feature-prediction error:

$$P_\Phi = \arg\min_{P_k} \sum_s \|P_k^T\Phi(s) - E\{\Phi(s'|s)\}\|_2^2. \tag{4}$$

(For brevity, we shall henceforth leave $s' \sim P(s'|s)$ implicit). One way to solve the minimization problem in Eq. (4) is to compute the expected next feature explicitly as the $n \times k$ matrix $P\Phi$ and then find the least-squares solution to the over-constrained system $\Phi P_\Phi \approx P\Phi$, since the i^{th} row of ΦP_Φ is P_Φ's prediction of the next feature values for state i and the i^{th} row of $P\Phi$ is the expected value of these features. The least-squares solution is

$$P_\Phi = (\Phi^T\Phi)^{-1}\Phi^T P\Phi, \tag{5}$$

with approximate next feature values $\widehat{P\Phi} = \Phi P_\Phi$. To predict the reward model using the same features, we could perform a standard least-squares projection into $span(\Phi)$ to compute an approximate reward predictor:

$$r_\Phi = (\Phi^T\Phi)^{-1}\Phi^T R, \tag{6}$$

with corresponding approximate reward: $\hat{R} = \Phi r_\Phi$. As in the value-function approximation case, it is possible to do a weighted L_2 projection, a straightforward generalization that we omit for conciseness of presentation.

Classically, an advantage of learning a model and deriving values from the model (indirect learning) over using samples to estimate the values (direct learning) is that such a method can be very data efficient. On the other hand, learning an accurate model can require a great deal of experience. Surprisingly, we find that the two approaches are the same, at least in the linear approximation setting.

3. Linear Fixed-Point Solution = Linear-Model Solution

The notion that linear fixed-point methods are implicitly computing some sort of model has been recognized in varying degrees for several years. For example, Boyan (1999)

[1] Permitting linearly dependent basis functions would not change our results, but would complicate exposition since the resulting weight vectors would no longer be unique. In practice, one may use SVD to enforce the selection of a unique solution when features are linearly dependent.

considered the intermediate calculations performed by LSTD in some special cases, and interpreted parts of the LSTD algorithm as computing a compressed model. In this section, we show that the linear fixed-point solution for features Φ is *exactly* the solution to the linear model described by P_Φ and r_Φ. We first prove it for the uncontrolled case, and then generalize our result to the controlled case. Our results concern unweighted projections, but generalize readily to weighted projections.

3.1. The Uncontrolled Case

Recall that the approximate model transforms feature vectors to feature vectors, so any k-vector is a state in the approximate model. If $\mathbf{x}$ is such a state, then, in the approximate model, ${r_\Phi}^T\mathbf{x}$ is the reward for this state and ${P_\Phi}^T\mathbf{x}$ is the next state vector. The Bellman equation for state $\mathbf{x}$ is:

$$V[\mathbf{x}] = {r_\Phi}^T\mathbf{x} + \gamma V[{P_\Phi}^T\mathbf{x}] = \sum_{i=0}^{\infty} \gamma^i {r_\Phi}^T ({P_\Phi}^i)^T\mathbf{x}.$$

Expressed with respect to the original state space, the value function becomes

$$V = \Phi \sum_{i=0}^{\infty} \gamma^i {P_\Phi}^i r_\Phi,$$

which is a linear combination of the columns of Φ. Since $V = \Phi\mathbf{w}$ for some $\mathbf{w}$, the fixed-point equation becomes:

$$\begin{aligned} V &= \hat{R} + \gamma\widehat{P\Phi}\mathbf{w} && (7)\\ \Phi\mathbf{w} &= \Phi r_\Phi + \gamma\Phi P_\Phi\mathbf{w} && (8)\\ \mathbf{w} &= (I - \gamma P_\Phi)^{-1} r_\Phi. && (9) \end{aligned}$$

We call the solution to the system above the *linear model solution*. A solution will exist when P_Φ has a spectral radius less than $1/\gamma$. This condition is not guaranteed because P_Φ is not necessarily a stochastic matrix; it is simply a matrix that predicts expected next feature values. The cases where the spectral radius of P_Φ exceeds $1/\gamma$ correspond to the cases where the value function defined by P_Φ and r_Φ assigns unbounded value to some states.

Theorem 3.1 *For any MRP M and set of features Φ, the linear-model solution and the linear fixed-point solution are identical.*

Proof We begin with the expression for the linear-model solution from Eq. (9) and then proceed by substituting the definitions of P_Φ and r_Φ from Eq. (5) and Eq. (6), yielding:

$$\begin{aligned} \mathbf{w} &= (I - \gamma P_\Phi)^{-1} r_\Phi \\ &= (I - \gamma(\Phi^T\Phi)^{-1}\Phi^T P\Phi)^{-1}(\Phi^T\Phi)^{-1}\Phi^T R \\ &= \mathbf{w}_\Phi. \ \blacksquare \end{aligned}$$

This result demonstrates that for a given set of features Φ, there is no difference between using the exact model to find an *approximate* linear fixed-point value function in terms of Φ and first constructing an approximate linear model in terms of Φ and then solving for the *exact* value function of the approximate model using Eq. (9). Although the model-based view produces exactly the same value function as the value-function-based view, the model-based view can give a new perspective on error analysis and feature selection, as shown in later sections.

3.2. The Controlled Case: LSPI

For the controlled case, we denote a *policy* as $\pi : S \mapsto A$. Since rewards and transitions are action dependent, the value function is defined over state–action pairs and is called a Q-function: For a fixed policy π, Q^π is the unique fixed-point solution to the Bellman equation

$$Q^\pi[s_i, a] = R_i^a + \gamma \sum_j P_{ij}^a Q^\pi[s_j, \pi(s_j)].$$

As in Section 2.1, Q^π can be approximated by functions in $span(\Phi)$: $\hat{Q} = \sum_{i=1}^k w_i\phi_i$, but now the basis functions ϕ_i are defined over state–action pairs rather than states.

In the controlled case, the policy π can be refined iteratively, as in the Least-Squares Policy Iteration (LSPI) algorithm (Lagoudakis & Parr, 2003). Starting with an arbitrary policy π_1, LSPI performs two steps iteratively until certain termination conditions are satisfied. In iteration i, it first computes an approximate linear value function $\hat{Q}_i$ for the current policy π_i (the *policy-evaluation* step), and then computes a new policy π_{i+1} that is greedy with respect to $\hat{Q}_i$ (the *policy-improvement* step).

In the policy-evaluation step, an algorithm LSTDQ, which is the Q-version of the LSTD algorithm, is used to compute $\hat{Q}_i$. Since a Markov decision process controlled by a fixed policy is equivalent to an induced Markov reward process whose state space is $S \times A$, LSTDQ can be viewed as LSTD running over this induced MRP. Due to Theorem 3.1, LSTDQ effectively builds a least-squares linear model approximation and then finds the *exact* solution to this model. Therefore, the intermediate value functions $\hat{Q}_i$ found by LSPI are the exact value functions of the respective approximate linear models with the smallest (weighted) L_2 error.

4. Analysis of Error: Uncontrolled Case

Value-function-based methods often analyze the error of a value function $\hat{V}$ in terms of the one-step lookahead error, or *Bellman error*:

$$BE(\hat{V}) = R + \gamma P\hat{V} - \hat{V}.$$

In the context of linear value functions and linear models, we shall define the Bellman error for a set Φ of features as the error in the linear fixed-point value function for Φ:

$$BE(\Phi) = BE(\Phi \mathbf{w}_\Phi) = R + \gamma P \Phi \mathbf{w}_\Phi - \Phi \mathbf{w}_\Phi.$$

To understand the relationship between the error in the linear model and the Bellman error, we define two components of the model error, the *reward error*:

$$\Delta_R = R - \hat{R},$$

and the *per-feature error*:

$$\Delta_\Phi = P\Phi - \widehat{P\Phi}.$$

The per-feature error is the error in the prediction of the expected next feature values, so both terms can be thought of as the residual error of the linear model. The next theorem relates the Bellman error to these model errors.

Theorem 4.1 *For any MRP M and features Φ,*

$$BE(\Phi) = \Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi. \tag{10}$$

Proof Using the definitions of $BE(\Phi)$, Δ_R, and Δ_Φ:

$$\begin{aligned} BE(\Phi) &= R + \gamma P \Phi \mathbf{w}_\Phi - \Phi \mathbf{w}_\Phi \\ &= (\Delta_R + \hat{R}) + \gamma(\Delta_\Phi + \widehat{P\Phi})\mathbf{w}_\Phi - \Phi \mathbf{w}_\Phi \\ &= (\Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi) + \hat{R} + (\gamma \Phi P_\Phi - \Phi)\mathbf{w}_\Phi \\ &= (\Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi) + \hat{R} - \Phi(I - \gamma P_\Phi)\mathbf{w}_\Phi \\ &= (\Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi) + \hat{R} - \Phi r_\Phi \\ &= \Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi. \end{aligned}$$

The final step follows from the definition of $\hat{R}$, and the penultimate step follows from Eq. (9) and Theorem 3.1. ∎

This decomposition of the Bellman error lets us think of the Bellman error as composed of two separate sources of error: reward error, and per-feature error. In the next section, we show that this view can give insight into the problem of feature selection, but we also caution that there can be interactions between Δ_R and Δ_Φ. For example, consider the basis composed of the single basis function $\Phi^* = [V^*]$. Clearly, $BE(\Phi^*) = 0$, but for any non-trivial problem and approximate model, Δ_R and Δ_Φ will be nonzero and will cancel each other out in Eq. (10).

A similar result may be possible for the controlled case, but there are some subtleties. For example, there is not a clean notion of a fixed point for the outer loop of the LSPI algorithm since the algorithm is not guaranteed to converge to a single policy or $\mathbf{w}$.

5. Feature Selection

We present several insights on the problem of feature selection that follow from the results presented above.

5.1. General Observations about Δ_R and Δ_Φ

The condition $\Delta_R = \Delta_\Phi = 0$ is sufficient (but not necessary) to achieve zero Bellman error and a perfect value function. Specifically, it requires that the features of the approximate model capture the structure of the reward function, and that the features of the approximate model are sufficient to predict expected next features. In the case where Φ is a set of indicator functions over disjoint partitions of S, these conditions are similar to those specified for model minimization (Dean & Givan, 1997) in MDPs.

Features that are insufficient to represent the immediate reward are likely to be problematic since any error in the prediction of the immediate reward based upon the features (Δ_R) can appear directly in the Bellman error through the first summand of Eq. (10). This finding is consistent with the observation of Petrik (2007) of the problems that arise when the reward is orthogonal to the features.

For $\Delta_\Phi = 0$, the Bellman error is determined entirely by Δ_R, with no dependence on γ. This observation has some interesting implications for feature selection and the analysis of the resulting approximate value function, topics we address further in Section 5.3.

5.2. Incremental Feature Generation

This section presents two existing methods for incrementally building a basis, the *Krylov basis*, and *Bellman Error Basis Functions* (BEBFs). We also propose a new method based upon the model error, Model Error Basis Functions (MEBFs), then show that all three methods are equivalent given the same initial conditions.

5.2.1. The Krylov Basis

The Krylov basis is defined in terms of powers of the transition matrix multiplied by R. We refer to the Krylov basis with k basis functions, starting from $\mathbf{X}$, as $Krylov_k(\mathbf{X})$, with $Krylov_k(\mathbf{X}) = \{P^{i-1}\mathbf{X} : 1 \leq i \leq k\}$. For an MRP, typically $\mathbf{X} = R$. The Krylov basis, and Krylov methods in general, are standard techniques for the iterative solution to systems of linear equations. Its relevance to feature selection for RL was demonstrated by Petrik (2007).

5.2.2. BEBFs

Many researchers have proposed using features based upon the residual error in the current feature set (Wu & Givan, 2004; Sanner & Boutilier, 2005; Keller et al., 2006). Parr et al. (2007) describe this family of techniques as Bellman Error Basis Functions (BEBFs), and analyze some of the properties of this approach. More formally, if $\Phi_k \mathbf{w}_{\Phi_k}$ is the current value function, BEBF adds $\phi_{k+1} = BE(\Phi_k)$ as the next basis function. We refer to the basis resulting from $k-1$ iterations of BEBF, starting from $\mathbf{X}$, as $BEBF_k(\mathbf{X})$.

Theorem 5.1 *(Petrik[2]) For any $k \geq 1$,*

$$span(Krylov_k(R)) = span(BEBF_k(R)).$$

Proof The proof is by induction on k. For the basis:

$$Krylov_1(R) = BEBF_1(R) = R.$$

For the inductive step, we assume equality up to k, so for both methods the value function can be expressed as:

$$\Phi_k \mathbf{w}_{\Phi_k} = \sum_{i=1}^{k} w_i P^{i-1} R.$$

Now, observe that:

$$BE(\Phi_k) = R + \gamma P(\sum_{i=1}^{k} w_i P^{i-1} R) - \sum_{i=1}^{k} w_i P^{i-1} R.$$

The only part of the above that is not already in the basis is the contribution from $P^{k+1}R$, which is precisely what is added in $Krylov_{k+1}(R)$. ∎

5.2.3. MEBFs

A natural generalization of BEBFs to the model-based view would be to add features that capture the residual error in the model. Starting from Φ_k, this technique (MEBF) adds Δ_R and Δ_Φ (or the linearly independent components thereof) to the basis at each iteration to create Φ_{k+1}. In contrast to BEBFs, this method can add a large number of basis functions at each iteration since Δ_Φ has as many columns as Φ. One might imagine that this process could result in an exponential growth in the number of basis functions. In fact, however, the number of new basis functions added at each iteration will not grow since each new set of basis functions that is added will drive the error in the previous basis functions to 0.

We refer to the basis resulting from $k-1$ iterations of MEBF, starting from $\mathbf{X}$, as $MEBF_k(\mathbf{X})$. For an initial basis of Φ, the MEBF basis expansion is guaranteed to contain the BEBF basis expansion.

Theorem 5.2 $span(BEBF_2(\Phi)) \subseteq span(MEBF_2(\Phi))$.

Proof Follows immediately from Eq. (10). ∎

Theorem 5.3 *For $k \geq 1$:*

$$span(Krylov_k(R)) = span(MEBF_k(R)).$$

Proof The proof is by induction on k. For the basis:

$$Krylov_1(R) = MEBF_1(R) = R.$$

[2] M. Petrik, personal communication, 2007.

For the inductive step, we assume equality up to k and consider the behavior of MEBF. For $k \geq 1$, $\Delta_R = 0$, since R is the first basis function added. The basis Φ_k is equivalent to a collection of basis functions of the form $\phi_i = P^{i-1}R$ for $1 \leq i \leq k$. As a result, $P\phi_i$ is already in the basis for all $1 \leq i < k$. Thus, the only nonzero column of Δ_Φ will correspond to feature ϕ_k and will be $P^k R - P_\Phi P^{k-1} R$. Since $P_\Phi P^{k-1} R$ is necessarily in $span(\Phi_k)$, the only new contribution to the basis made by MEBF will be from $P^k R$, which is precisely what is added by $Krylov_{k+1}(R)$. ∎

These results show that, starting from R, all three methods will produce the same basis. An advantage of BEBF is that it will produce orthogonal basis vectors. An advantage of MEBF is that it can add multiple new basis vectors at each iteration if it is initialized with a set of basis functions.

5.3. Invariant Subspaces of P

The form of the Bellman error in Eq. (10) suggests that features for which $\Delta_\Phi = 0$ are particularly interesting. If a dictionary of such features were readily available, then the feature-selection problem would reduce to the problem of predicting the immediate reward using this dictionary.

The condition $\Delta_\Phi = 0$ means that, collectively, the features are a basis for a perfect linear predictor of their *own* next, expected values. More formally, features Φ are *subspace invariant* with respect to P if $P\Phi$ is in $span(\Phi)$, which means that there exists a Λ such that $P\Phi = \Phi\Lambda$.

At first, it may seem like subspace invariance is an extraordinary requirement that could hold only for a complete basis for P. It turns out, however, that there are many ways to describe invariant subspaces of P. Any set of eigenvectors of P forms an invariant subspace. For eigenvectors $\mathcal{X}_1 \ldots \mathcal{X}_k$ with eigenvalues $\lambda_1 \ldots \lambda_k$, $\Lambda = \text{diag}(\lambda_1 \ldots \lambda_k)$. The set of generalized eigenvectors corresponding to a particular eigenvalue λ of P is subspace invariant with respect to P. For an eigenvalue λ with multiplicity i, there will be i generalized eigenvectors, $\mathcal{X}_1 \ldots \mathcal{X}_i$ satisfying $(P - \lambda I)\mathcal{X}_j = \mathcal{X}_{j-1}$ for $1 \leq j \leq i$ and $(P - \lambda I)^j \mathcal{X}_j = 0$ for $0 \leq j \leq i$. More generally, if Φ^1 and Φ^2 are subspace invariant with respect to P, then so is their union. Finally, the Schur decomposition of a matrix P provides a set of nested invariant subspaces of P.

In fairness, we point out that these methods all require knowledge of P and superlinear computation time in the dimension of P. We defer discussion of the practicality of implementing these methods to Section 6 and Section 7.

Theorem 5.4 *For any MRP M and subspace invariant feature set Φ, $\Delta_\Phi = 0$.*

Proof First, we observe that P_Φ has a particularly simple

form as a consequence of subspace invariance:

$$P_\Phi = (\Phi^T\Phi)^{-1}\Phi^T P\Phi = (\Phi^T\Phi)^{-1}\Phi^T\Phi\Lambda = \Lambda.$$

Substituting into the definition of Δ_Φ:

$$\Delta_\Phi = P\Phi - \widehat{P\Phi} = P\Phi - \Phi P_\Phi = \Phi\Lambda - \Phi\Lambda = 0. \quad \blacksquare$$

Subspace invariant features have additional intriguing properties. The resulting value function always exists and can be interpreted as the result of using the true transition model with the approximate reward function $\hat{R}$.

Theorem 5.5 *For any MRP M and subspace invariant feature set Φ, $\mathbf{w}_\Phi$ always exists and*

$$\Phi\mathbf{w}_\Phi = (I - \gamma P)^{-1}\hat{R}.$$

Proof Starting with the form of $\mathbf{w}_\Phi$ from Eq. (7) and the fact that $\Delta_\Phi = 0$:

$$\begin{aligned} \Phi\mathbf{w}_\Phi &= \hat{R} + \gamma\widehat{P\Phi}\mathbf{w}_\Phi \\ &= \hat{R} + \gamma P\Phi\mathbf{w}_\Phi \\ \Phi\mathbf{w}_\Phi - \gamma P\Phi\mathbf{w}_\Phi &= \hat{R} \\ \Phi\mathbf{w}_\Phi &= (I - \gamma P)^{-1}\hat{R}. \end{aligned}$$

To confirm that such a $\mathbf{w}_\Phi$ actually exists, we note that $\hat{R} \in span(\Phi)$ by construction, and that $(I - \gamma P)^{-1}$ must exist for the actual P and $0 \leq \gamma < 1$, allowing us to rewrite:

$$\Phi\mathbf{w}_\Phi = \sum_{i=0}^{\infty} \gamma^i P^i \hat{R},$$

which remains in $span(\Phi)$ because of Φ's subspace invariance with respect to P. $\blacksquare$

Our analysis has some similarities with that of Petrik (2007). Petrik considered the eigenvalue decomposition of P as a basis and considered the error in the projection of V^* into this basis. Petrik also suggested the use of the Jordan form, which would provide generalized eigenvectors for matrices that are not diagonalizable. Our analysis focuses on the Bellman error of the linear fixed-point solution. Insights from the model-based view of linear approximation architectures allow us to decompose the error into distinct components corresponding to the reward and transition models, making the role of invariant subspaces particularly salient.

6. Experimental Results

We present policy-evaluation results on three different problems. Our objective is to demonstrate how our theoretical results can inform the feature-selection process and explain the behavior of known feature-selection algorithms. We consider 4 algorithms:

PVF: This is the proto-value function (PVF) framework described by Mahadevan and Maggioni (2007). PVFs use eigenvalues of the Laplacian derived from an empirically constructed adjacency matrix (from random walk trajectories), enumerated in increasing order of eigenvalue. We reproduced their method as closely as possible, including adding links to the adjacency matrix for all policies, not just the policy under evaluation. Curiously, removing the off-policy links seemed to produce worse performance. We avoided using samples to eliminate the confounding (for our purposes) issue of variance between experiments. We used the combinatorial Laplacian for the 50-state and blackjack problems, but used the normalized Laplacian in the two-room problem to match Mahadevan and Maggioni (2007).

PVF-MP: This algorithm selects basis functions from the set of PVFs, but selects them incrementally based upon the Bellman error. Specifically, basis function $k+1$ is the PVF that has highest dot product with the Bellman error resulting from the previous k basis functions. It can be interpreted as a form of matching pursuits (Mallat & Zhang, 1993) on the Bellman error with a dictionary of PVFs.

Eig-MP: This algorithm is similar to PVF-MP, but selects from a dictionary of the eigenvectors of P. Both Eig-MP and PVF-MP are similar in spirit to Petrik's WL algorithm.

BEBF: This is the BEBF algorithm starting with $\Phi_0 = R$, as described in Section 5.2.

Our experiments performed unweighted L_2 projection and report unweighted L_2 norm error. We also considered L_∞ error and L_2 projections weighted by stationary distributions, but the results were not qualitatively different. We report the Bellman error, the reward error, and the *feature error*, which is the contribution of the per-feature errors to the Bellman error: $\gamma\Delta_\Phi\mathbf{w}_\Phi$. These metrics are presented as a function of the number of basis functions.

6.1. 50-state Chain

We applied all 4 algorithms to the 50-state chain problem from Lagoudakis and Parr (2003), with the results shown in Figure 1(a–c). As demanded by theory, Eig-MP has 0 feature error, which means that the entirety of the Bellman error is expressed in Δ_R. BEBFs represent the other extreme since $\Delta_R = 0$ after the first basis function is added and the entirety of the Bellman error is expressed through Δ_Φ. For this problem, PVFs appear to be approximately subspace invariant, resulting in low Δ_Φ. However, both Eig-MP and the PVF methods do poorly because the reward is not easily expressed as linear combination of a small number of PVFs. PVF-MP does better than plain PVFs because it is actively trying to reduce the error, while plain PVFs choose basis functions in an order that ignores the reward.

6.2. Two-room Problem

We tried all four algorithms on an optimal policy for the two-room navigation problem from Mahadevan and Maggioni (2007). The transition matrix for this problem is not diagonalizable and typical methods for extracting generalized eigenvectors proved unreliable, so we do not show results for the Eig-MP method. Figure 1(d–f) shows the breakdown of error for the remaining algorithms. In this case, the Laplacian approach produces features that behave less like an invariant subspace, resulting in high Δ_R and Δ_Φ. However, there is some cancellation between them.

6.3. Blackjack

We tested a version of the bottomless-deck blackjack problem from Sutton and Barto (1998), evaluating the policy they propose. For the model described in the book, all methods except BEBF performed extremely poorly. To make the problem more amenable to eigenvector-based methods, we implemented an ergodic version that resets to an initial distribution over hands with a value of 12 or larger and used a discount of 0.999. The breakdown of error for the different algorithms is shown in Figure 1(g–i), where we again omit Eig-MP. As expected, BEBFs exhibit $\Delta_R = 0$, and drive the Bellman error down fairly rapidly. PVFs exhibit some interesting behaviors: When the PVFs are enumerated in order of increasing eigenvalue, they form an invariant subspace. As a result, the feature error for PVFs hugs the abscissa in Figure 1(i). However, this ordering completely fails to match R until the very last eigenvectors are added, resulting in very poor performance overall. In contrast, PVF-MP adds basis eigenvectors in an order that does not result in subspace invariant features sets, but that does match R earlier, resulting in a more consistent reduction of error.

7. Discussion and Future Work

A significant finding in our work is the close relationship between value-function approximation and model-based learning. Our experimental results illustrate the relationship between the power of the features to represent an approximate model and the Bellman error.

While features that represent feature transitions accurately have highly desirable properties, both components of the model, the reward function and the transition function, should be respected by the features. Both a strength and weakness of the BEBF/MEBF/Krylov methods is their connection to specific policies and reward structures. Our results are consonant with those of Petrik (2007), which showed good performance for the Krylov basis and some surprisingly weak performance for eigenvector-based methods despite their appealing properties.

To focus on the expressive power of the features, our results in this paper do not directly consider sampled data, the regime in which linear fixed-point methods are most often employed. Some initial results on the effects of noise in feature generation for BEBF/MEBF/Krylov methods can be found in Parr et al. (2007), however further analysis would still be helpful. For eigenvector-based methods, there are some questions about the cost of estimating eigenvectors of P, or an approximation to P via the Laplacian. Computing eigenvectors can be computationally intensive and, for general P, prone to numerical instabilities.

An important direction for future work is seeking a deeper understanding of the interaction between feature-selection and policy-improvement algorithms such as LSPI.

8. Conclusion

This paper demonstrated a fundamental equivalence between linear value-function approximation and linear model approximation for RL. This equivalence led to a novel view of the Bellman error, which then gave insight into the problem of feature selection. These insights were used to explain the behavior of existing feature-selection algorithms on some sample problems. While this research has not, yet, led to a novel algorithmic approach, we believe that it helps address fundamental questions of representation and feature selection encountered by anyone wishing to solve real RL problems.

Acknowledgment

We thank Carlo Tomasi and Xiaobai Sun for helpful discussions, Sridhar Mahadevan and Jeff Johns for pointing out some discrepancies between our interpretation of the two-room problem in an earlier version of this paper and the version in Mahadevan and Maggioni (2007), and Marek Petrik for Theorem 5.1. This work was supported in part by DARPA CSSG HR0011-06-1-0027, and by NSF IIS-0713435. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

References

Boyan, J. A. (1999). Least-squares temporal difference learning. *ICML-99*.

Bradtke, S., & Barto, A. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, *2*.

Dean, T., & Givan, R. (1997). Model minimization in Markov decision processes. *AAAI-97*.

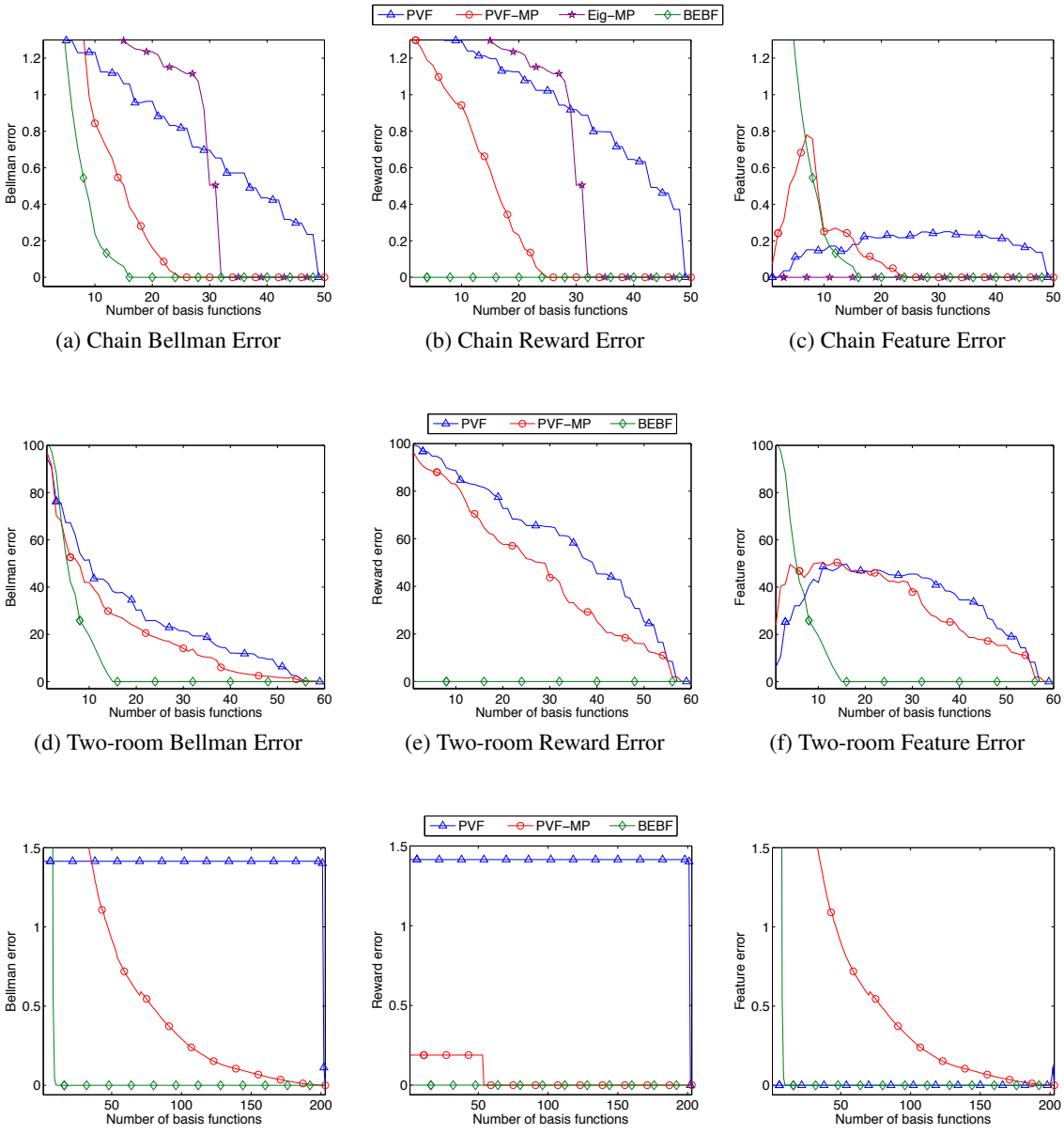
Keller, P., Mannor, S., & Precup, D. (2006). Automatic basis function construction for approximate dynamic programming and reinforcement learning. *ICML 2006*.

Koller, D., & Parr, R. (1999). Computing factored value functions for policies in structured MDPs. *IJCAI-99*.

Lagoudakis, M., & Parr, R. (2003). Least squares policy iteration. *JMLR*, *4*.

Mahadevan, S., & Maggioni, M. (2007). Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *JMLR*, *8*.

Mallat, S. G., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Trans. on Signal Processing*, *41*.



(a) Chain Bellman Error (b) Chain Reward Error (c) Chain Feature Error

(d) Two-room Bellman Error (e) Two-room Reward Error (f) Two-room Feature Error

(g) Ergodic Blackjack Bellman Error (h) Ergodic Blackjack Reward Error (i) Ergodic Blackjack Feature Error

Figure 1. Decomposition of the Bellman error for three different problems. First row: 50-state chain; Second row: Two-room problem; Third row: Ergodic Blackjack. First column: Bellman error; Second Column: reward error; Third Column: feature error

Parr, R., Painter-Wakefield, C., Li, L., & Littman, M. (2007). Analyzing feature generation for value-function approximation. *ICML-07*.

Petrik, M. (2007). An analysis of Laplacian methods for value function approximation in MDPs. *IJCAI-07*.

Sanner, S., & Boutilier, C. (2005). Approximate linear programming for first-order MDPs. *UAI-05*.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. The MIT Press.

Wu, J.-H., & Givan, R. (2004). *Feature-discovering approximate value iteration methods* (Technical Report TR-ECE-04-06). Purdue University.

Yu, H., & Bertsekas, D. (2006). *Convergence results for some temporal difference methods based on least squares* (Technical Report LIDS-2697). MIT.