Kernelized Value Function Approximation for Reinforcement Learning

Gavin Taylor GVTAYLOR@CS.DUKE.EDU
Ronald Parr PARR@CS.DUKE.EDU
Department of Computer Science, Duke University, Durham, NC 27708 USA

Keywords: Reinforcement Learning

Abstract

A recent surge in research in kernelized approaches to reinforcement learning has sought to bring the benefits of kernelized machine learning techniques to reinforcement learning. Kernelized reinforcement learning techniques are fairly new and different authors have approached the topic with different assumptions and goals. Neither a unifying view nor an understanding of the pros and cons of different approaches has yet emerged. In this paper, we offer a unifying view of the different approaches to kernelized value function approximation for reinforcement learning. We show that, except for different approaches to regularization, Kernelized LSTD (KLSTD) is equivalent to a model-based approach that uses kernelized regression to find an approximate reward and transition model, and that Gaussian Process Temporal Difference learning (GPTD) returns a mean value function that is equivalent to these other approaches. We also discuss the relationship between our model-based approach and the earlier Gaussian Processes in Reinforcement Learning (GPRL). Finally, we decompose the Bellman error into the sum of transition error and reward error terms, and demonstrate through experiments that this decomposition can be helpful in choosing regularization parameters.

1. Introduction

Kernelized reinforcement learning methods seek to bring the benefits of kernelized machine learning techniques to value function approximation in reinforcement learning (RL). Several different approaches to kernelized RL (KRL) have been proposed in recent years. Some of these approaches were motivated via Gaussian processes (Engel et al., 2005; Rasmussen & Kuss, 2004) and replacement of the covariance function with an arbitrary kernel. It is also possible to take a more direct approach by replacing the dot product in a standard RL algorithm, such as LSTD (Xu et al., 2005). Even though there is great similarity in the motivation for these different approaches, the derivations are quite different and the proposed algorithms are not obviously equivalent.

In this paper we propose a unifying view of KRL. We present a novel kernelized transition model and reward function approximation that is derived from kernelized regression. We then show that, except for regularization, the value function obtained by solving the approximate model is identical to that of Kernelized LSTD (KLSTD) (Xu et al., 2005). For a particular choice of regularization parameters these solutions are also identical to the mean returned by Gaussian Process Temporal Difference Learning (GPTD) (Engel et al., 2005). Moreover, the earlier, model-based Gaussian Processes in Reinforcement Learning (GPRL) (Rasmussen & Kuss, 2004) is also equivalent to these other methods, with minor differences resulting from the choice of kernel and regularization parameters.

Even though these methods are all fundamentally alike, they offer different viewpoints on the problem and it may be advantageous to adopt different views in different situations. For example, GP based methods also provide covariances which may be useful to guide exploration. Our novel model-based approach has the benefit of permitting separate regularization terms for the transition model and reward model. Following Parr et al. (2008), we provide a decomposition of the Bellman error into a sum of reward and transitions errors. This decomposition can be useful for the model-based view because it can guide the selection of potentially different regularization parameters for each component of the solution. We provide a simple experiment to demonstrate this use of the Bellman error decomposition.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

The focus of this paper is value function approximation for a fixed policy, and not policy improvement. Our results apply to pure prediction problems, but also to the policy evaluation step that occurs in the inner loop of policy iteration algorithms that use kernelized RL for policy evaluation (Xu et al., 2007; Rasmussen & Kuss, 2004). The only place where our results are incompatible with policy improvement may be in our suggestion to use the Bellman error decomposition for parameter tuning. It is conceivable, though unlikely in our opinion, that kernel and regularization parameters that are good for one policy could be bad for another. In this case, our approach could still be used, but the kernel or regularization parameters might need to be re-tuned for each policy.

Our goal in this work is to provide a unifying view that encompasses many uses of kernels in reinforcement learning, but some methods do not fit neatly into this framework. Extremely recent work by Farahmand et al. (2008) introduces a regularized, kernelized version of LSTD. This has many similarities with the techniques discussed herein, but uses a larger kernel matrix which includes kernel values between pairs of next states, making it not directly comparable to earlier methods. This work focuses on the use of kernels in value function approximation, but kernelized machine learning techniques have also been used to represent policies in policy search methods, e.g., Lagoudakis and Parr (2003), or Bagnell and Schneider (2003).

2. Formal Framework and Notation

This work concerns uncontrolled Markov processes, referred to as Markov reward processes (MRP): $M = (S, P, R, \gamma)$. Given a state $s_i \in S$, the probability of a transition to a state s_j is given by P_{ij} and results in an expected reward of r_i. If the policy for a Markov Decision Process (MDP) is treated as a constant, then the MDP induces an MRP. The task of computing the value function for the resulting MRP is sometimes referred to as policy evaluation.

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 or closely approximate the solution to the Bellman equation:

$$V^* = R + \gamma P V^*.$$

For any matrix, A, we use A^T to indicate the transpose of A. Additionally, we use $\mathbf{r}$ to represent a vector of sampled rewards.

3. Kernel-Based Value Function Approximators

We begin by defining kernels and kernelized regression. A *kernel* is a symmetric function between two points, denoted $k(x_i, x_j) = k(x_j, x_i)$. A kernel matrix, $\mathbf{K}$, stores kernel values for all pairs in a dataset with $K_{ij} = K_{ji} = k(\mathbf{x}_i, \mathbf{x}_j)$, If this $\mathbf{K}$ is positive semi-definite, Mercer's theorem states that the kernel function can be interpreted as a dot product between the two points in a higher-dimensional space. The use of this property to increase the expressiveness of the feature space without explicitly adding more features is known as the *kernel trick*. Kernel-based methods can be viewed as means of easing the machine learning practitioner's feature engineering burden. However, the expressiveness of kernels comes with a risk of overfitting and a potential for heavy computation despite the efficiency of the kernel trick.

If regularized least-squares regression is re-derived using the kernel trick, we arrive at the dual (kernelized) form of linear least-squares regression (Bishop, 2006),

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T \left(\mathbf{K} + \mathbf{\Sigma}\right)^{-1} \mathbf{t}, \tag{1}$$

where $\mathbf{t}$ represents the target values of the sampled points, and $\mathbf{k}(\mathbf{x})$ is a column vector with elements $k_i(\mathbf{x}) = k(\mathbf{x}_i, \mathbf{x})$. $\mathbf{\Sigma}$ is a generic regularization term. Frequently $\mathbf{\Sigma} = \lambda \mathbf{I}$, but as in general Tikhonov regression, non-zero off-diagonal terms are possible. The choice of the variable $\mathbf{\Sigma}$ to represent the regularization term is no coincidence. As in ordinary Tikhonov regression, the regularizer plays an equivalent role to a noise covariance in Bayesian regression. This connection between the regularizer and the noise covariance in the Bayesian solution carries over to the Gaussian Process (GP) regression case where the mean of the GP solution is identical to Equation 1.

Kernelized regression techniques (and kernelized value function approximation techniques) require inverting a matrix that is as large as the kernel matrix itself, with a number of entries that is quadratic in the training set size. This has led numerous authors to propose sparsification techniques in general, e.g. Engel et al. (2002), and for RL algorithms, e.g. Engel et al. (2005). We view sparsification as a general technique for approximating the result that would have been obtained if the original $\mathbf{K}$ had been used, so our theoretical results compare only the original versions of algorithms without sparsification.

4. Kernelized Value Function Approximation

In this section of the paper, we review two approaches to direct, kernelized value function approximation. These algorithms are direct in the sense that they do not explicitly construct a model, though we will later show in section 5 that the solutions they produce are equivalent to those produced by model-based approaches.

4.1. KLSTD

Kernel-based least-squares temporal difference learning (KLSTD) (Xu et al., 2005) begins with the general LSTD(λ) (Boyan, 1999) algorithm and uses the kernel trick to derive a kernelized version of LSTD. For brevity, we focus on the $\lambda = 0$ case, for which the approximate value function solution is

$$\hat{V}(s) = \mathbf{k}(s)^T \left(\mathbf{KHK}\right)^{-1} \mathbf{Kr}, \tag{2}$$

where

$$\mathbf{H} = \begin{bmatrix} 1 & -\gamma & 0 & \dots & 0 \\ 0 & 1 & -\gamma & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & -\gamma \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

KLSTD was presented without any form of regularization, but the authors did use sparsification which may have helped regularize their solution somewhat and may have also helped improve the conditioning of $\mathbf{K}$ in their experimental results.

4.2. GPTD

Gaussian Process Temporal Difference Learning (GPTD) (Engel et al., 2005) takes a GP approach to direct value function approximation solution. They begin by modeling the residual:

$$R(s) = V(s) - \gamma V(s') + N(s, s').$$

N is modeled with a Gaussian process, $N(s, s') \sim \mathcal{N}(0, \mathbf{\Sigma})$. This formulation results in the approximate value function represented as a Gaussian distribution. The mean of this distribution is

$$\hat{V}(s) = \mathbf{k}(s)^T \mathbf{H}^T \left(\mathbf{HKH}^T + \mathbf{\Sigma}\right)^{-1} \mathbf{r}, \tag{3}$$

where $\mathbf{H}$ is defines as in KLSTD.[1] As with KLSTD, they propose a sparsification technique that approximates the above solution with less computation.

5. Kernel-Based Models

This section presents two approaches to kernelized reinforcement learning that first produce kernelized approximations of the transition and reward models, and then solve for the value function of the approximate transition and reward models. The first approach is Rassmussen and Kuss's Gaussian processes in reinforcement learning (GPRL) (2004). The second is a novel method based directly upon kernelized regression.

[1] GPTD actually defined $\mathbf{H}$ without the last row of KLSTD's $\mathbf{H}$. The last row corresponds to the assumption that the trajectory ends with a transition to an absorbing state, an assumption we preserve for our version of GPTD for the convenience of having a square $\mathbf{H}$.

5.1. GPRL

In GPRL, transitions and value functions are modeled with a Gaussian process. Additionally, kernels are assumed to be the sum of a Gaussian kernel and a weighted delta function; the resulting kernel matrix is denoted $\mathbf{K}_v$. Rewards are assumed to be noiseless. This construction makes the value function

$$\hat{V}(s) = R(s) + \gamma \int P_{s_i, s'} \hat{V}(s') ds'$$

difficult to calculate in closed form, as $P_{s_i,s'}$ and $\hat{V}(s')$ are both Gaussian. Using a result from Girard et al. (2003), GPRL approximates $\int P_{s_i,s'} V(s') ds'$ with $\mathbf{W}_i \mathbf{K}_v^{-1} \mathbf{V}$, replacing the integral over the entire state space with a product of $\mathbf{W}_i$, which is the expected next kernel values given state s_i, and the kernelized representation of the value function, $\mathbf{K}_v^{-1}\mathbf{V}$. The final result is a Gaussian distribution for the value of all sampled points with mean

$$\hat{\mathbf{V}} = \left(\mathbf{I} - \gamma \mathbf{W} \mathbf{K}_v^{-1}\right)^{-1} \mathbf{r},$$

where $\mathbf{W}$ is a matrix of expected next kernel values with $\mathbf{W}_{ij} = \mathbb{E}\left[k(s_i', s_j)\right]$. Because the kernel function consists of a sum, $\mathbf{K}_v$ can be decomposed such that $\mathbf{K}_v = \mathbf{K} + \sigma^2 \mathbf{\Delta}$, where $\mathbf{K}$ is the kernel matrix resulting from the Gaussian kernel, and $\mathbf{\Delta}_{ij} = \delta(i, j)$, producing the value function

$$\hat{\mathbf{V}} = \left(\mathbf{I} - \gamma \mathbf{W}(\mathbf{K} + \sigma^2 \mathbf{\Delta})^{-1}\right)^{-1} \mathbf{r}. \tag{4}$$

5.2. A General Kernelized Model-Based Solution

We now present a general, kernelized approach to model-based RL built upon kernelized regression. Using equation 1 with $\mathbf{\Sigma} = \mathbf{0}$, we can formulate our unregularized *approximate reward model* for state s as

$$\hat{R}(s) = \mathbf{k}(s)^T \mathbf{K}^{-1} R, \tag{5}$$

and the regularized version:

$$\hat{R}(s) = \mathbf{k}(s)^T \left(\mathbf{K} + \mathbf{\Sigma}_R\right)^{-1} R. \tag{6}$$

The *approximate transition model* is similar. Our approach differs from GPRL in one important way: GPRL learns an approximate model that predicts next state variables given current state variables. Our approximate model does not seek to predict base features, but seeks to predict kernel values, i.e, $\mathbf{k}(s')$ given $\mathbf{k}(s)$[2]. This defines the relationship of our predicted next state to our sampled points in the space implicitly defined by the kernel function. We first define the matrix $\mathbf{K}' = P\mathbf{K}$, in which $K'_{ij} = \mathbb{E}\left[k(x'_i, x_j)\right]$. Here, $\mathbf{K}'$ can be thought of as target data consisting of the vectors $\mathbf{k}(s')$. To approximate the transition model, we again use kernelized regression:

$$\hat{\mathbf{k}}(s') = \mathbf{k}(s)^T \mathbf{K}^{-1}\mathbf{K}'. \tag{7}$$

As with the reward model, we can construct a regularized version of the approximate transition model,

$$\hat{\mathbf{k}}(s') = \mathbf{k}(s)^T \left(\mathbf{K} + \mathbf{\Sigma}_P\right)^{-1} \mathbf{K}' \tag{8}$$

We can use the models expressed in equations 5 and 7 to construct an unregularized approximate value function.

$$\begin{aligned}\hat{V}(s) =& \mathbf{k}(s)^T\mathbf{K}^{-1}R + \gamma \underbrace{\underbrace{\mathbf{k}(s)^T\mathbf{K}^{-1}\mathbf{K}'}_{\hat{\mathbf{k}}(s')^T}\mathbf{K}^{-1}R}_{\hat{R}(s')} + \\ & \gamma^2 \mathbf{k}(s)^T \left(\mathbf{K}^{-1}\mathbf{K}'\right)^2 \mathbf{K}^{-1}R + \dots \\ =& \mathbf{k}(s)^T \sum_{i=0}^{\infty} \left[\gamma^i \left(\mathbf{K}^{-1}\mathbf{K}'\right)^i\right] \mathbf{K}^{-1}R \\ =& \mathbf{k}(s)^T \left(\mathbf{I} - \gamma\mathbf{K}^{-1}\mathbf{K}'\right)^{-1} \mathbf{K}^{-1}R\end{aligned}$$

Distributing $\mathbf{K}^{-1}$, we arrive at our final value function,

$$V(s) = \mathbf{k}(s)^T \left(\mathbf{K} - \gamma\mathbf{K}'\right)^{-1} R. \tag{9}$$

It is also possible to perform this derivation using the regularized reward and model approximations from equations 6 and 8, resulting in the following value function:

$$\hat{V}(s) = \mathbf{k}(s)^T \left[\left(\mathbf{K} + \mathbf{\Sigma}_R\right) - \gamma\left(\mathbf{K} + \mathbf{\Sigma}_R\right)\left(\mathbf{K} + \mathbf{\Sigma}_P\right)^{-1}\mathbf{K}'\right]^{-1} R \tag{10}$$

Regularization is often necessary, as real RL problems exhibit noise and the high expressiveness of the kernel matrix can result in overfitting. Also, there is no guarantee the kernel matrix $\mathbf{K}$ will be invertible and regularization tends to improve the conditioning of the matrix inversion problem. A benefit of the model-based solution presented here is that it offers the ability to regularize reward and transition approximations separately.

[2]This approach is similar to the one taken by Parr et al.(2008), where an approximate model was used to predict next feature value given current feature values.

So far, our derivation has assumed that $\mathbf{K}'$ could be computed and represented explicitly for the entire state space. In practice one would typically apply kernel-based approximation algorithms to large or continuous state spaces. In these cases, it is impossible to create a kernel matrix $\mathbf{K}$ representing the kernel functions between every state. Instead, we sample (s, r, s') triples from the state space, and construct a *sampled* $\mathbf{K}$, $\mathbf{K}'$, and $\mathbf{r}$. In this case, $k(s_i, s_j)$ is the kernel between the starting states in the i^{th} and j^{th} triple in the data set, and $\mathbf{K}'_{ij}$ contains kernel values between the i^{th} next state and the j^{th} starting state in the data set. In the special case where the samples are drawn from trajectories, $s'_i = s_{i+1}$, and $k(s'_i, s_j) = k(s_{i+1}, s_j)$. Therefore, $\mathbf{K}' = \mathbf{GK}$, where

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}.$$

Additionally, $\mathbf{H} = \mathbf{I} - \gamma\mathbf{G}$.

6. Equivalence

With the model-based value functions defined, we are ready to state our equivalence theorems.

Theorem 6.1 *Given the same trajectories and same kernels, the KLSTD value function is equivalent to the unregularized model-based value function.*

Proof Our first step is to show that $\mathbf{KHK} = \mathbf{KK} - \gamma\mathbf{KK}'$, using $\mathbf{H} = \mathbf{I} - \gamma\mathbf{G}$ and $\mathbf{K}' = \mathbf{GK}$:

$$\begin{aligned}\mathbf{KHK} =& \mathbf{K}\left(\mathbf{I} - \gamma\mathbf{G}\right)\mathbf{K} \\ =& \mathbf{KK} - \gamma\mathbf{KGK} \\ =& \mathbf{KK} - \gamma\mathbf{KK}'.\end{aligned}$$

Starting from the solution to KLSTD in equation 2,

$$\begin{aligned}V(s) =& \mathbf{k}(s)^T \left(\mathbf{KHK}\right)^{-1} \mathbf{Kr} \\ =& \mathbf{k}(s)^T \left(\mathbf{KK} - \gamma\mathbf{KK}'\right)^{-1} \mathbf{Kr} \\ =& \mathbf{k}(s)^T \left(\mathbf{K}^{-1}\mathbf{KK} - \gamma\mathbf{K}^{-1}\mathbf{KK}'\right)^{-1} \mathbf{r} \\ =& \mathbf{k}(s)^T \left(\mathbf{K} - \gamma\mathbf{K}'\right)^{-1} \mathbf{r},\end{aligned}$$

which is equal to our unregularized approximate model-based value function in equation 9. ∎

The implication of this theorem is that we can view KLSTD as implicitly approximating the underlying transition and reward functions of the system. We can prove a similar theorem about the relationship of GPTD to the model-based approximation.

Theorem 6.2 *Given the same trajectories and same kernels, the mean value function returned by GPTD is equivalent to the regularized model-based value function with* $\boldsymbol{\Sigma}_R = \boldsymbol{\Sigma}_P = \boldsymbol{\Sigma}(\mathbf{H}^T)^{-1}$.

Proof We begin with the GPTD mean approximate value function introduced in equation 3, and show it is equivalent to equation 10:

$$\begin{aligned}\hat{V}(s) =&\mathbf{k}(s)^T\mathbf{H}^T\left(\mathbf{HKH}^T+\boldsymbol{\Sigma}\right)^{-1}\mathbf{r}\\ =&\mathbf{k}(s)^T\left(\mathbf{HK}+\boldsymbol{\Sigma}(\mathbf{H}^T)^{-1}\right)^{-1}\mathbf{r}\\ =&\mathbf{k}(s)^T\left(\mathbf{K}-\gamma\mathbf{K}'+\boldsymbol{\Sigma}(\mathbf{H}^T)^{-1}\right)^{-1}\mathbf{r}\\ =&\mathbf{k}(s)^T\left[(\mathbf{K}+\boldsymbol{\Sigma}_R)-\gamma\left(\mathbf{K}+\boldsymbol{\Sigma}_R\right)\left(\mathbf{K}+\boldsymbol{\Sigma}_P\right)^{-1}\mathbf{K}'\right]^{-1}\mathbf{r},\end{aligned}$$

when $\boldsymbol{\Sigma}_R = \boldsymbol{\Sigma}_P = \boldsymbol{\Sigma}(\mathbf{H}^T)^{-1}$. ∎

In the noiseless case when $\boldsymbol{\Sigma} = \mathbf{0}$, GPTD is equivalent to the unregularized model-based value function, equation 9.

This theorem assumes that $(\mathbf{H}^T)^{-1}$ exists, but this is ensured by the structure of $\mathbf{H}$. As with KLSTD, we see that GPTD can be viewed as implicitly approximating the underlying transition and reward models of the system. It is not surprising that GPTD's $\boldsymbol{\Sigma}$ appears in both the transition model and reward regularization since GPTD does not have separate noise terms for the reward and transition. The appearance of $(\mathbf{H}^T)^{-1}$ may be somewhat surprising. Loosely speaking, we can say that it propagates the regularizer $\boldsymbol{\Sigma}$ through the transition model since:

$$(\mathbf{H}^T)^{-1} = (I-\gamma P^T)^{-1} = \sum_{i=1}^{\infty}\gamma^i(P^T)^i,$$

but we believe that empirical study of the role of $(\mathbf{H}^T)^{-1}$ is warranted.

In contrast, GPRL explicitly models the transition model of the system, but using a very different starting point from our model-based approximation. However, we can show the final results are still closely related.

Theorem 6.3 *Using the same sample data and same kernel functions, the mean value function returned by GPRL is equivalent to the regularized model-based value function with* $\boldsymbol{\Sigma}_R = \boldsymbol{\Sigma}_P = \sigma^2\boldsymbol{\Delta}$.

Proof Recall that $\mathbf{W}_{ij} = \mathbb{E}\left[k(x_i', x_j)\right]$. GPRL's regularization term is part of the definition of the kernel, but GPRL uses a trick to ensure that regularization is applied only to the training data and not to test data: the regularizer is multiplied by a delta function which ensures that it is applied only to on-diagonal entries in $\mathbf{K}$. Since GPRL assumes that data are drawn from a Gaussian, $P(\delta(x_i', x_j) > 0) = 0$, and σ^2 is not expected to appear in $\mathbf{K}'$. We therefore assume $\mathbf{W}_{ij} = \mathbf{K}'_{ij}$. Beginning with the GPRL value function in equation 4:

$$\begin{aligned}\hat{\mathbf{V}} =& \left(\mathbf{I}-\gamma\mathbf{W}(\mathbf{K}+\sigma^2\boldsymbol{\Delta})^{-1}\right)^{-1}\mathbf{r}\\ =&\left(\mathbf{I}-\gamma\mathbf{K}'(\mathbf{K}+\sigma^2\boldsymbol{\Delta})^{-1}\right)^{-1}\mathbf{r}\\ =&(\mathbf{K}+\sigma^2\boldsymbol{\Delta})\left((\mathbf{K}+\sigma^2\boldsymbol{\Delta})-\gamma\mathbf{K}'(\mathbf{K}+\sigma^2\boldsymbol{\Delta})^{-1}(\mathbf{K}+\sigma^2\boldsymbol{\Delta})\right)^{-1}\mathbf{r}\\ =&(\mathbf{K}+\sigma^2\boldsymbol{\Delta})\left(\mathbf{K}+\sigma^2\boldsymbol{\Delta}-\gamma\mathbf{K}'\right)^{-1}\mathbf{r}.\end{aligned}$$

To change this value function from one that defines approximate values over all experienced points to one that defines a value function on an arbitrary point s, we replace $\mathbf{K}+\sigma^2\boldsymbol{\Delta}$ with the kernel evaluated at s:

$$\hat{V}(s) = \left(\mathbf{k}(s)^T+\sigma^2\delta(s)^T\right)\left(\mathbf{K}+\sigma^2\boldsymbol{\Delta}-\gamma\mathbf{K}'\right)^{-1}\mathbf{r},$$

where $\delta(s)$ is a column vector with elements $\delta_i(s) = \delta(s_i, s)$. For an arbitrary point in a continuous space, $\delta(s)$ will be the zero vector, so

$$\begin{aligned}\hat{V}(s) =&\mathbf{k}(s)\left(\mathbf{K}+\sigma^2\boldsymbol{\Delta}-\gamma\mathbf{K}'\right)^{-1}\mathbf{r}\\ =&\mathbf{k}(s)^T\left[(\mathbf{K}+\boldsymbol{\Sigma}_R)-\gamma\left(\mathbf{K}+\boldsymbol{\Sigma}_R\right)\left(\mathbf{K}+\boldsymbol{\Sigma}_P\right)^{-1}\mathbf{K}'\right]^{-1}\mathbf{r},\end{aligned}$$

when $\boldsymbol{\Sigma}_R = \boldsymbol{\Sigma}_P = \sigma^2\boldsymbol{\Delta}$. ∎

When $\sigma^2 = 0$, this is equivalent to the unregularized model-based value function in equation 9.

In summary, KLSTD, GPTD, GPRL, and our novel model-based value function differ only in their approaches to regularization and their assumptions about the manner in which the samples are drawn. Even though these algorithms are basically identical, it can nevertheless be useful to consider different approaches to regularization to get insight into parameter selection since overfitting with kernel methods is a genuine concern. This section also shows that even seemingly direct approaches to kernelized value function approximation have a model-based interpretation. In subsequent sections, we show that the model-based interpretation lends itself to a useful error decomposition.

7. Analysis of Error

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

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

The Bellman error captures how well an approximation predicts the value of the next state given the value of the current state, and bounds the actual error of the approximation:

$$\| V^* - \hat{V} \|_\infty \leq \frac{\| BE(\hat{V}) \|_\infty}{1-\gamma}.$$

In the linear approximation case that the Bellman error can be broken down into components that express the approximation's error in estimating the process's reward and transition functions (Parr et al., 2008). Similarly, in kernel-based cases, the Bellman error can be decomposed into *reward error* and *transition error*. Reward error expresses how well the approximation represents the reward function, while transition error expresses how well the approximation represents the transition function of the system. As was the case in the linear approximation domain, this view can provide insight into the performance of the approximation algorithm.

To be precise about the error over the entire state space, we must distinguish between the states that have been sampled, and the rest of the states. We learn a value function for the entire state space containing $|S|$ states using n sampled points, where $n \ll |S|$. Thus, $\mathbf{r}$ is an n-vector, and $\hat{P}$ and $\mathbf{K}$ are $n \times n$ matrices, where $\hat{P}$ is our estimation of P. Let $\mathcal{K}$ be a $|\mathcal{S}| \times n$ matrix, where $\mathcal{K}_{ij}$ is equal to $k(s_i, s_j)$, where s_i is any element of S, and s_j is a sampled point. The approximate reward values for the entire of the state space follow from equation 5:

$$\hat{R} = \mathcal{K}(\mathbf{K} + \mathbf{\Sigma}_R)^{-1}\mathbf{r}. \tag{11}$$

We can also introduce *reward error* in the kernel context as a vector indicating the difference between the actual reward values, and our approximation $\hat{R}$.

$$\Delta_R = R - \hat{R} \tag{12}$$

Similarly, we introduce *transition error*[3]. In the kernel context, this is a measure of our error in predicting $\mathbf{k}(s')$ given arbitrary $s \in S$:

$$\Delta_{\mathbf{K}'} = P\mathcal{K} - \mathcal{K}(\mathbf{K} + \mathbf{\Sigma}_P)^{-1}\hat{P}\mathbf{K} \tag{13}$$

For convenience, we define

$$\mathbf{w} = \left(I - \gamma(\mathbf{K} + \mathbf{\Sigma}_P)^{-1}\hat{P}\mathbf{K}\right)^{-1}(\mathbf{K} + \mathbf{\Sigma}_R)^{-1}\mathbf{r},$$

which is extracted from our value function defined in Equation 9.

Theorem 7.1 *The Bellman error of a KRL value function approximation is a linear combination of reward and transition errors.*

[3]We thank Sridhar Mahadevan for pointing out that Bertsekas and Castanon (1989) introduce a similar concept in a somewhat different context.

Proof For brevity, we show only the unregularized case with $\mathbf{\Sigma}_P = \mathbf{\Sigma}_R = 0$; the regularized derivation parallels this one. For any MRP M and kernel matrix $\mathbf{K}$, the approximate value function over the entire state space is $\mathcal{K}\mathbf{w}$. Starting with the definition of Bellman error:

$$\begin{aligned}
&BE(\mathcal{K}\mathbf{w}) \\
&\quad = R + \gamma P\mathcal{K}\mathbf{w} - \mathcal{K}\mathbf{w} \\
&\quad = \Delta_R + \hat{R} + \gamma\left(\Delta_{\mathbf{K}'} + \mathcal{K}\mathbf{K}^{-1}\hat{P}\mathbf{K}\right)\mathbf{w} - \mathcal{K}\mathbf{w} \\
&\quad = \Delta_R + \hat{R} + \gamma\Delta_{\mathbf{K}'}\mathbf{w} + \gamma\mathcal{K}\mathbf{K}^{-1}\hat{P}\mathbf{K}\mathbf{w} - \mathcal{K}\mathbf{w} \\
&\quad = \Delta_R + \hat{R} + \gamma\Delta_{\mathbf{K}'}\mathbf{w} + \mathcal{K}\left(\gamma\mathbf{K}^{-1}\hat{P}\mathbf{K} - \mathbf{I}\right)\mathbf{w} \\
&\quad = \Delta_R + \hat{R} + \gamma\Delta_{\mathbf{K}'}\mathbf{w} + \mathcal{K}\mathbf{K}^{-1}\left(\gamma\hat{P}\mathbf{K} - \mathbf{K}\right)\mathbf{w} \\
&\quad = \Delta_R + \hat{R} + \gamma\Delta_{\mathbf{K}'}\mathbf{w} - \mathcal{K}\mathbf{K}^{-1}\left(\mathbf{K} - \gamma\hat{P}\mathbf{K}\right)\mathbf{w} \\
&\quad = \Delta_R + \hat{R} + \gamma\Delta_{\mathbf{K}'}\mathbf{w} - \mathcal{K}\mathbf{K}^{-1}\left(\mathbf{K} - \gamma\hat{P}\mathbf{K}\right)\left(\mathbf{K} - \gamma\hat{P}\mathbf{K}\right)^{-1}\mathbf{r} \\
&\quad = \Delta_R + \hat{R} + \gamma\Delta_{\mathbf{K}'}\mathbf{w} - \mathcal{K}\mathbf{K}^{-1}\mathbf{r} \\
&\quad = \Delta_R + \hat{R} + \gamma\Delta_{\mathbf{K}'}\mathbf{w} - \hat{R}
\end{aligned}$$

At this point, we arrive at our result,

$$BE(\mathcal{K}\mathbf{w}) = \Delta_R + \gamma\Delta_{\mathbf{K}'}\mathbf{w}. \quad \blacksquare \tag{14}$$

This result means that when using kernel-based methods we can view the Bellman error as having two direct sources of error, the reward error and the transition error. This is useful because the reward error and transition error can be estimated easily from a testing or cross validation set; this makes the process of selecting kernels or kernel parameters fairly transparent since model and reward fitting are straightforward (kernelized) regression problems. In contrast, tuning value function approximators based upon an approximate value function is a notoriously opaque process that requires expertise in MDPs.

8. Experimental Results

We have shown several kernel-based value function approximators to be equivalent, except for their treatment of regularization. The Bellman error decomposition provides insight into the behavior of regularization and can facilitate the selection of regularizing parameters.

To demonstrate this use of the Bellman error decomposition, we consider a continuous, two-room maze navigation problem similar to problems explored by Mahadevan and Maggioni(2006) and Engel et al.(2005). A vertical wall separates the state space into equal-sized left and right rooms. A small opening in the middle of this wall acts as a passage between the rooms. An agent can be at any empty point. The actions are 1-unit long steps in one of the 4 cardinal directions, with added two-dimensional Gaussian noise with covariance $0.5\mathbf{I}$. The agent cannot travel through walls, so movements that would cross walls are truncated at the point of intersection. When the agent reaches a 1-unit-wide goal region along the entire right wall of the right room, it deterministically receives a reward of 1. In our version of the problem, the agent can loiter in the reward region, making the maximum achievable value $1/(1-\gamma)$. We used $\gamma = 0.9$.

Our training set consisted of 3750 random (s, r, s') samples drawn uniformly from the space. The action used at each state was from an optimal policy for this problem. We used a Gaussian kernel with a diagonal covariance matrix $3\mathbf{I}$, and varied the $\mathbf{\Sigma}_P$ regularization term to demonstrate the effects on the value function and Bellman error decomposition. Figures 1(a), 1(b), 1(c), and 1(d) show the approximate value function and absolute value of the error components at the grid points resulting for $\mathbf{\Sigma}_R = \mathbf{\Sigma}_P = 0$. We used SVD to handle the ill-conditioned matrix inversion problems that resulted in this case. Note that the scale of the error plots is different from the scale of the value function plot. The large errors edges of the space are due to our sampling strategy, which provided fewer neighboring points at the edges and corners. More interestingly, however, the value function is extremely uneven and inconsistent throughout the space. It is difficult to identify why by looking only at the Bellman error in figure 1(b). However, it is clear from examining 1(c) that the Bellman error is dominated by the transition error component and that there is overfitting across much of the domain. However, the reward error in figure 1(d) is much less uneven, indicating that, the reward approximation is not overfitting. This is not surprising since the reward is deterministic for this problem. Overall, these results are consistent with a need for greater regularization in the approximate transition function. This is precisely what one would expect with priori knowledge of a noisy transition function and deterministic rewards; in practice, however, we would not have a priori knowledge of the noise sources in the model.

Figures 1(e) and 1(f) show the value function and transition error resulting from the same approximation, but with $\mathbf{\Sigma}_P = 0.1\mathbf{I}$. Reward error is not shown as it remained unchanged. As we expected, the value function is smoother. Figure 1(f) shows the transition error of the regularized approximation. Note again that the scale of the error plot is different from the scale of the value function plot. There are still areas of high error, as the large amount of noise relative to the number of samples still makes an approximation difficult. However, the transition error is much better controlled, demonstrating the regularization has performed as we might expect. Overall, the value function has a far more reasonable shape and the remaining problems with the value function are primarily attributable to a scarcity of samples in the corners.

In this particular example, the Bellman error decomposition is less useful for revealing an excess of regularization. Under the optimal policy for this problem, the agent does not bump into any walls, except in cases where it is caused by noise. The transition function is therefore a fairly smooth function and excessive regularization has the effect of making the approximation too flat. This has a global effect of squashing the value function, but, in contrast with the case of too little regularization, the effect is not particularly salient from a graph of the transition error alone.

This simple example demonstrates both the advantage of separate reward and transition regularizers, and the insight offered by the Bellman error decomposition. From a wider perspective, the model-based viewpoint decouples the transition and reward approximation aspects of the problem, and the Bellman error decomposition provides a window into understanding the behavior of otherwise opaque approximation schemes.

9. Conclusions

In this work, we first demonstrated that several "model-free" kernelized value function approximators can be viewed as special cases of a novel, model-based value function approximator. These conclusions showed that direct value-function approximators were still approximating the model, an insight that can allow kernelized value-function approximation to be reduced to two straightforward kernelized regression problems, one for the transition model and one for the reward model.

Additionally, we demonstrated that the Bellman error resulting from a kernelized approximator can be viewed as a linear combination of errors in the approximation of the reward function and errors in approximation of the transition function. This can be leveraged to understand the previously opaque behavior of kernelized value function approximators and help tune kernel parameters.

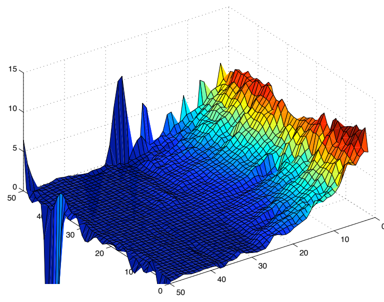
By presenting a unified view of kernelized RL and reducing the kernelized RL problem to two more manageable kernelized regression problems, we hope to make it easier for practitioners to understand, implement, and achieve success with kernelized RL.

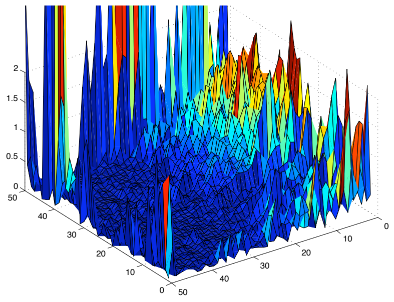
Acknowledgments

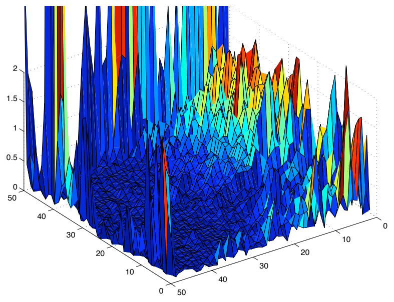
This work was supported in part by DARPA CSSG HR0011-06-1-0027, and by NSF IIS-0713435. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

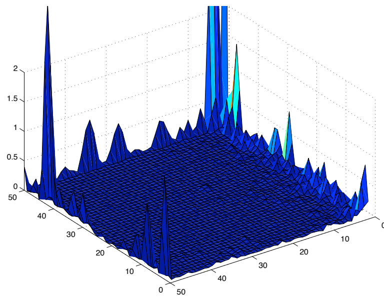
References

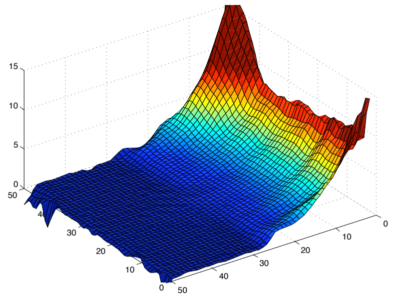
Bagnell, J. A. D., & Schneider, J. (2003). *Policy Search in Reproducing Kernel Hilbert Space* (Technical Report


(a) Value function with $\mathbf{\Sigma}_P = \mathbf{0}$


(b) Bellman error with $\mathbf{\Sigma}_P = \mathbf{0}$


(c) Transition error with $\mathbf{\Sigma}_P = \mathbf{0}$


(d) Reward error with $\mathbf{\Sigma}_P = \mathbf{0}$


(e) Value function with $\mathbf{\Sigma}_P = 0.1\mathbf{I}$

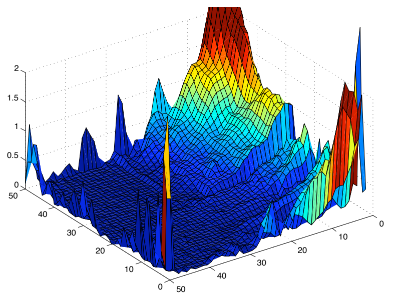

(f) Transition error with $\mathbf{\Sigma}_P = 0.1\mathbf{I}$

Figure 1. Bellman error decomposition for the continuous two-room problem, with two different values for the transition regularization matrix $\mathbf{\Sigma}_p$. Note that the scale on the value function plots is different from the scale on the error plots.

CMU-RI-TR-03-45). Robotics Institute, Pittsburgh, PA.

Bertsekas, D. P., & Castanon, D. A. (1989). Adaptive Aggregation Methods for Infinite Horizon Dynamic Programming. *IEEE Transactions on Automatic Control* (pp. 589–598).

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Boyan, J. A. (1999). Least-Squares Temporal Difference Learning. *In Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 49–56). Morgan Kaufmann.

Engel, Y., Mannor, S., & Meir, R. (2002). Sparse Online Greedy Support Vector Regression. *13th European Conference on Machine Learning* (pp. 84–96).

Engel, Y., Mannor, S., & Meir, R. (2005). Reinforcement Learning with Gaussian Processes. *Machine Learning-International Workshop then Conference* (pp. 201–208).

Farahmand, A. M., Ghavamzadeh, M., Szepesvari, C., & Mannor, S. (2008). Regularized Policy Iteration. *Advances in Neural Information Processing Systems* (pp. 441–448).

Girard, A., Rasmussen, C. E., Candela, J. Q., & Murray-Smith, R. (2003). Gaussian Process Priors with Uncertain Inputs-Application to Multiple-Step Ahead Time Series Forecasting. *Advances in Neural Information Processing Systems* (pp. 545–552).

Lagoudakis, M. G., & Parr, R. (2003). Reinforcement Learning as Classification: Leveraging Modern Classifiers. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 424–431).

Mahadevan, S., & Maggioni, M. (2006). *Proto-value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes* (Technical Report). University of Massachusetts.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., & Littman, M. (2008). An Analysis of Linear Models, Linear Value-Function Approximation, and Feature Selection for Reinforcement Learning. *International Conference of Machine Learning* (pp. 752–759).

Rasmussen, C. E., & Kuss, M. (2004). Gaussian Processes in Reinforcement Learning. *Advances in Neural Information Processing Systems* (pp. 751–759).

Xu, X., Hu, D., & Lu, X. (2007). Kernel-Based Least Squares Policy Iteration for Reinforcement Learning. *IEEE Transactions on Neural Networks* (pp. 973–992).

Xu, X., Xie, T., Hu, D., & Lu, X. (2005). Kernel Least-Squares Temporal Difference Learning. *International Journal of Information Technology* (pp. 54–63).