Approximate dynamic programming using model-free Bellman Residual Elimination

2010 American Control Conference
Marriott Waterfront, Baltimore, MD, USA
June 30-July 02, 2010

Approximate Dynamic Programming Using Model-Free Bellman Residual Elimination

Brett Bethke and Jonathan P. How

Abstract— **This paper presents an modification to the method of Bellman Residual Elimination (BRE) [1], [2] for approximate dynamic programming. While prior work on BRE has focused on learning an approximate policy for an underlying Markov Decision Process (MDP) when the state transition model of the MDP is known, this work proposes a model-free variant of BRE that does not require knowledge of the state transition model. Instead, state trajectories of the system, generated using simulation and/or observations of the real system in operation, are used to build stochastic approximations of the quantities needed to carry out the BRE algorithm. The resulting algorithm can be shown to converge to the policy produced by the nominal, model-based BRE algorithm in the limit of observing an infinite number of trajectories. To validate the performance of the approach, we compare model-based and model-free BRE against LSPI [3], a well-known approximate dynamic programming algorithm. Measuring performance in terms of both computational complexity and policy quality, we present results showing that BRE performs at least as well as, and sometimes significantly better than, LSPI on a standard benchmark problem.**

I. Introduction

Markov Decision Processes (MDPs) are a powerful and general framework for addressing problems involving sequential decision making under uncertainty [4]. Unfortunately, since MDPs suffer from the well-known curse of dimensionality, the computational complexity of finding an exact solution is typically prohibitively large. Therefore, in most cases, approximate dynamic programming (ADP) techniques must be employed to compute approximate solutions that can be found in reasonable time [5].

Motivated by the success of kernel-based methods such as support vector machines [6] and Gaussian processes [7] in pattern classification and regression applications, researchers have begun applying these powerful techniques in the ADP domain. This work has led to algorithms such as GPTD [8], an approach which uses temporal differences to learn a Gaussian process representation of the cost-to-go function, and GPDP [9], which is an approximate value iteration scheme based on a similar Gaussian process cost-to-go representation. Another recently-developed approach, known as Bellman Residual Elimination (BRE) [1], [2], uses kernel-based regression to solve a system of Bellman equations over a small set of sample states.

Similar to the well-studied class of linear architectures [3], [10], kernel-based cost representations can be interpreted as mapping a state of the MDP into a set of features; however, unlike linear architectures, the effective feature vector of a kernel-based representation may be infinite-dimensional. This property gives kernel methods a great deal of flexibility and makes them particularly appropriate in approximate dynamic programming, where the structure of the cost function may not be well understood. The prior work on BRE [1], [2] has demonstrated that by taking advantage of this flexibility, a class of algorithms can be developed that enjoy several advantageous theoretical properties, including the ability to compute cost-to-go functions whose Bellman residuals are identically zero at the sampled states. This property, in turn, immediately implies that the BRE algorithms are guaranteed to converge to the optimal policy in the limit of sampling the entire state space. In addition to these theoretical properties, we have recently demonstrated encouraging results in applying BRE in order to compute approximate policies for large-scale, multi-agent UAV planning problems [11].

The BRE work to date has assumed that the underlying state transition model of the MDP is available, and has focused on developing a class of algorithms with the stated theoretical properties that can use this model information efficiently. Of course, in some circumstances, an exact model may not be known if the system in question is not well understood. To cope with these situations, this paper presents a model-free variation on the BRE algorithms developed to date. Instead of relying on knowledge of the state transition model, the new model-free approach uses state trajectory data, gathered either by simulation of the system or by observing the actual system in operation, to approximate the quantities needed to carry out BRE.

The paper begins by presenting background material on model-based BRE. It then shows how to modify the basic BRE approach when only state trajectory data, and not a full system model, is available. Several proofs demonstrate the correctness of the model-free BRE approach, establishing that the model-free algorithm converges to the same result as the model-based algorithm as more and more trajectory data is used. Finally, results of a comparison study between BRE and LSPI, a well-known approximate dynamic programming algorithm [3], show that both model-based and model-free BRE perform at least as well as, and sometimes significantly better than, LSPI on a standard benchmark problem.

II. Background

A. Markov Decision Processes

This paper considers the general class of infinite horizon, discounted, finite state MDPs. The MDP is specified by $(\mathcal{S}, \mathcal{A}, P, g)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space,

B. Bethke, Ph.D. bbethke@mit.edu
J. P. How is a Professor of Aeronautics and Astronautics, MIT, jhow@mit.edu

$P_{ij}(u)$ gives the transition probability from state i to state j under action u, and $g(i,u,j)$ gives the cost of moving from state i to state j under action u. To derive model-based BRE, we assume that the MDP model (i.e. the data $(\mathcal{S},\mathcal{A},P,g)$) is known. Future costs are discounted by a factor $0<\alpha<1$. A policy of the MDP is denoted by $\mu:\mathcal{S}\rightarrow\mathcal{A}$. Given the MDP specification, the problem is to minimize the cost-to-go function J_μ over the set of admissible policies Π:

$$\min_{\mu\in\Pi} J_\mu(i_0)=\min_{\mu\in\Pi}\mathbb{E}\left[\sum_{k=0}^{\infty}\alpha^k g(i_k,\mu(i_k))\right].$$

For notational convenience, the cost and state transition functions for a fixed policy μ are defined as

$$\begin{aligned} g_i^\mu &\equiv \sum_{j\in\mathcal{S}} P_{ij}(\mu(i))g(i,\mu(i),j) \qquad (1)\\ P_{ij}^\mu &\equiv P_{ij}(\mu(i)), \end{aligned}$$

respectively. The cost-to-go for a fixed policy μ satisfies the Bellman equation [4]

$$J_\mu(i)=g_i^\mu+\alpha\sum_{j\in\mathcal{S}}P_{ij}^\mu J_\mu(j)\quad\forall i\in\mathcal{S}, \qquad (2)$$

which can also be expressed compactly as $J_\mu=T_\mu J_\mu$, where T_μ is the (fixed-policy) dynamic programming operator.

B. Bellman Residual Elimination

Bellman Residual Elimination is a approximate policy iteration technique that is closely related to the class of Bellman residual methods [12]–[14]. Bellman residual methods attempt to perform policy evaluation by minimizing an objective function of the form

$$\left(\sum_{i\in\widetilde{\mathcal{S}}}|\widetilde{J}_\mu(i)-T_\mu\widetilde{J}_\mu(i)|^2\right)^{1/2}, \qquad (3)$$

where $\widetilde{J}_\mu$ is an approximation to the true cost function J_μ and $\widetilde{\mathcal{S}}\subset\mathcal{S}$ is a set of representative sample states. BRE uses a flexible kernel-based cost approximation architecture to construct $\widetilde{J}_\mu$ such that the objective function given by Eq. (3) is identically zero.

We now present the basic derivation of the BRE approach; for more details, see [1], [2]. To begin, BRE assumes the following functional form of the cost function:

$$\widetilde{J}_\mu(i)=\langle\underline{\Theta},\underline{\Phi}(i)\rangle. \qquad (4)$$

Here, $\langle\cdot,\cdot\rangle$ denotes the inner product in a Reproducing Kernel Hilbert Space (RKHS) [15] with the pre-specified kernel $k(i,i')=\langle\underline{\Phi}(i),\underline{\Phi}(i')\rangle$. $\underline{\Phi}(i)$ is the *feature mapping* of the kernel $k(i,i')$, and $\underline{\Theta}$ is a weighting element to be found by the BRE algorithm. The main insight behind BRE is the observation that the individual Bellman residuals $BR(i)$, given by

$$BR(i)\equiv\widetilde{J}_\mu(i)-T_\mu\widetilde{J}_\mu(i), \qquad (5)$$

can be simultaneously forced to zero by formulating a regression problem, which can then be solved using standard kernel-based regression schemes such as support vector regression [6] or Gaussian process regression [7]. To construct the regression problem, we substitute the functional form of the cost, given by Eq. (4), into Eq. (5):

$$BR(i)=\langle\underline{\Theta},\underline{\Phi}(i)\rangle-\left(g_i^\mu+\alpha\sum_{j\in\mathcal{S}}P_{ij}^\mu\langle\underline{\Theta},\underline{\Phi}(j)\rangle\right).$$

By exploiting linearity of the inner product $\langle\cdot,\cdot\rangle$, we can express $BR(i)$ as

$$\begin{aligned} BR(i) &= -g_i^\mu+\langle\underline{\Theta},\left(\underline{\Phi}(i)-\alpha\sum_{j\in\mathcal{S}}P_{ij}^\mu\underline{\Phi}(j)\right)\rangle\\ &= -g_i^\mu+\langle\underline{\Theta},\underline{\Psi}(i)\rangle, \end{aligned}$$

where $\underline{\Psi}(i)$ is defined as

$$\underline{\Psi}(i)\equiv\underline{\Phi}(i)-\alpha\sum_{j\in\mathcal{S}}P_{ij}^\mu\underline{\Phi}(j). \qquad (6)$$

By the preceding construction, every cost function $\widetilde{J}_\mu$ is associated with a function

$$\widetilde{W}_\mu\equiv\langle\underline{\Theta},\underline{\Psi}(i)\rangle \qquad (7)$$

which is closely related to the Bellman residuals through the relation $\widetilde{W}_\mu(i)=BR(i)+g_i^\mu$. Therefore, the condition that the Bellman residuals are identically zero at the sample states $\widetilde{\mathcal{S}}$ is equivalent to finding a function $\widetilde{W}_\mu$ satisfying

$$\widetilde{W}_\mu(i)=g_i^\mu\quad\forall i\in\widetilde{\mathcal{S}}. \qquad (8)$$

Eq. (7) implies that $\widetilde{W}_\mu$ belongs to a RKHS whose kernel is given by

$$\begin{aligned} \mathcal{K}(i,i') &= \langle\underline{\Psi}(i),\underline{\Psi}(i')\rangle \qquad (9)\\ &= \langle\ \underline{\Phi}(i)-\alpha\sum_{j\in\mathcal{S}}P_{ij}^\mu\underline{\Phi}(j)\ ,\\ &\qquad \underline{\Phi}(i')-\alpha\sum_{j\in\mathcal{S}}P_{i'j}^\mu\underline{\Phi}(j)\ \rangle \qquad (10) \end{aligned}$$

$\mathcal{K}(i,i')$ is called the *Bellman kernel associated with* $k(i,i')$. Solving the regression problem given by Eq. (8), using the associated Bellman kernel and any kernel-based regression technique (such as support vector regression, Gaussian process regression, etc), yields the weighting element $\underline{\Theta}$. By the representer theorem [16, 4.2], $\underline{\Theta}$ can be expressed as a linear combination of features $\underline{\Psi}$ of the sample states:

$$\underline{\Theta}=\sum_{i'\in\widetilde{\mathcal{S}}}\lambda_{i'}\underline{\Psi}(i').$$

Therefore, the output of solving the kernel regression problem is compactly described by the coefficients $\{\lambda_{i'}\}_{i'\in\widetilde{\mathcal{S}}}$. Once these coefficients are known, Eqs. (4) and (6) can be

used to compute the approximate cost function $\widetilde{J}_\mu$:

$$\begin{aligned}\widetilde{J}_\mu(i) &= \langle \underline{\Theta}, \underline{\Phi}(i) \rangle \\ &= \Big\langle \sum_{i' \in \widetilde{\mathcal{S}}} \lambda_{i'} \underline{\Psi}(i') \ , \ \underline{\Phi}(i) \Big\rangle \\ &= \sum_{i' \in \widetilde{\mathcal{S}}} \lambda_{i'} \left(k(i', i) - \alpha \sum_{j \in \mathcal{S}} P^\mu_{i'j} k(j, i) \right). \end{aligned} \tag{11}$$

To summarize, the preceding construction has shown how to perform policy evaluation for a fixed policy μ, given a base kernel $k(i, i')$ and a set of sample states $\widetilde{\mathcal{S}}$. The steps to perform approximate policy iteration are as follows:

1) Solve the regression problem [Eq. (8)] using the associated Bellman kernel $\mathcal{K}(i, i')$ [Eq. (10)] and any kernel-based regression technique to find the coefficients $\{\lambda_{i'}\}_{i' \in \widetilde{\mathcal{S}}}$.
2) Use the coefficients found in step 1 to compute the cost function $\widetilde{J}_\mu$ [Eq. (11)].
3) Perform a policy improvement step based on the cost function found in step 2, and return to step 1 to evaluate this new policy.

Under a technical, nondegeneracy condition on the kernel $k(i, i')$ (which is satisfied by many commonly-used kernels including radial basis function kernels), it can be proven that the Bellman residuals of the resulting cost function $\widetilde{J}_\mu$ are always identically zero at the states $\widetilde{\mathcal{S}}$, as desired [2]. As a result, it is straightforward to show that BRE is guaranteed to yield the *exact* cost J_μ in the limit of sampling the entire state space ($\widetilde{\mathcal{S}} = \mathcal{S}$).

Graph Structure of the Associated Bellman Kernel: It is useful to examine the associated Bellman kernel $\mathcal{K}(i, i')$ in some detail to understand its structure. Eq. (9) shows that $\mathcal{K}(i, i')$ can be viewed as the inner product between the feature mappings $\underline{\Psi}(i)$ and $\underline{\Psi}(i')$ of the input states i and i', respectively. In turn, Eq. (6) shows that $\underline{\Psi}(i)$ represents a new feature mapping that takes into account the local graph structure of the MDP, since it is a linear combination of $\underline{\Phi}$ features of both the state i and all of the successor states j that can be reached in a single step from i (these are all the states j for which P^μ_{ij} are nonzero).

Figure 1 is a graphical depiction of the associated Bellman kernel. Using the figure, we can interpret the associated Bellman kernel as measuring the total "overlap" or similarity between i, i', and all immediate (one-stage) successor states of i and i'. In this sense, the associated Bellman kernel automatically accounts for the local graph structure in the state space.

Computational Complexity

The computational complexity of model-based BRE is dominated by two factors: first, computing $\mathcal{K}(i, i')$ over every pair of sample states $(i, i') \in \widetilde{\mathcal{S}} \times \widetilde{\mathcal{S}}$ (this information is often called the *Gram matrix* of the kernel), and second, solving the regression problem. As illustrated in Fig. (1), computing $\mathcal{K}(i, i')$ involves enumerating each successor state of both i and i' and evaluating the base kernel $k(\cdot, \cdot)$ for each pair of successor states. If β is the average branching factor of the MDP, each state will have β successor states on average, so computing $\mathcal{K}(i, i')$ for a single pair of states (i, i') involves $\mathcal{O}(\beta^2)$ operations. Therefore, if $n_s \equiv |\widetilde{\mathcal{S}}|$ is the number of sample states, computing the full Gram matrix costs $\mathcal{O}(\beta^2 n_s^2)$ operations. The cost of solving the regression problem clearly depends on the regression technique used, but typical methods such as Gaussian process regression involve solving a linear system in the dimension of the Gram matrix, which involves $\mathcal{O}(n_s^3)$ operations. Thus, the total complexity of BRE is of order

$$\mathcal{O}(\beta^2 n_s^2 + n_s^3).$$

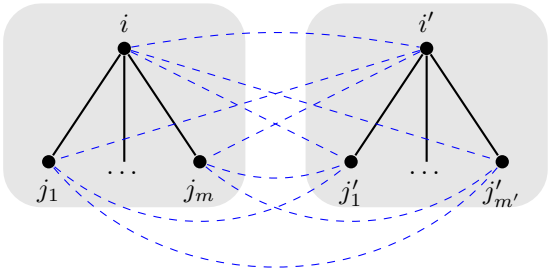

Fig. 1: Graph structure of the associated Bellman kernel $\mathcal{K}(i, i')$ [Eq. (9)]. The one-stage successor states of i and i' are $\{j_1, \ldots, j_m\}$ and $\{j'_1, \ldots, j'_{m'}\}$, respectively. The associated Bellman kernel is formed by taking a linear combination of the base kernel $k(\cdot, \cdot)$ applied to all possible state pairings (shown by the dashed blue lines) where one state in the pair belongs to the set of i and its descendants (this set is shown by the left shaded rectangle), and the other belongs to the set of i' and its descendants (right shaded rectangle).

III. Model-Free BRE

The BRE algorithm presented in the previous section requires knowledge of the system dynamics model, encoded in the probabilities P^μ_{ij}, to evaluate the policy μ. These probabilities are used in the calculation of the cost values g^μ_i [Eq. (1)], the associated Bellman kernel $\mathcal{K}(i, i')$ [Eq. (10)], and the approximate cost-to-go $\widetilde{J}_\mu(i)$ [Eq. (11)]. In particular, in order to solve the BRE regression problem given by Eq. (8), it is necessary to compute the values g^μ_i for $i \in \widetilde{\mathcal{S}}$, as well as the Gram matrix $\mathbb{K}$ of the associated Bellman kernel, defined by

$$\mathbb{K}_{ii'} = \mathcal{K}(i, i') \qquad i, i' \in \widetilde{\mathcal{S}}.$$

By exploiting the model information, BRE is able to construct cost-to-go solutions $\widetilde{J}_\mu(i)$ for which the Bellman residuals are exactly zero.

However, there may be situations in which it is not possible to use system model information directly. Clearly, one such situation is when an exact system model is unknown or unavailable. In this case, one typically assumes instead that a *generative model* of the system is available, which is a "black box" simulator that can be used to sample state trajectories of the system under a given policy. Alternatively,

one may collect state trajectory data from the actual system in operation and use this data in lieu of an exact model.

To address these situations, we now develop a variant of the BRE algorithm that uses only simulated trajectory data, obtained from simulations or from observing the real system, instead of using data from the true underlying system model P_{ij}^{μ}. This variant of BRE therefore represents a true, model-free, reinforcement learning algorithm. The key idea behind model-free BRE is to simulate a number of trajectories of length n, starting from each of the sample states in $\widetilde{\mathcal{S}}$, and use this information to build stochastic approximations to the cost values g_i^{μ}, kernel Gram matrix $\mathbb{K}$, and cost-to-go $\widetilde{J}_{\mu}(i)$. We use the notation T_l^{iq} to denote the l^{th} state encountered in the q^{th} trajectory starting from state $i \in \widetilde{\mathcal{S}}$, where l ranges from 0 to n (the length of the trajectory), and q ranges from 1 to m (the number of trajectories starting from each state $i \in \widetilde{\mathcal{S}}$).

Using the trajectory data T_l^{iq}, stochastic approximations of each of the important quantities necessary to carry out BRE can be formed. First, examining Eq. (1), notice that the cost values g_i^{μ} can be expressed as

$$\begin{aligned} g_i^{\mu} &= \sum_{j \in \mathcal{S}} P_{ij}(\mu(i)) g(i, \mu(i), j) \\ &\approx \frac{1}{m} \sum_{q=1}^{m} g(T_0^{iq}, \mu(T_0^{iq}), T_1^{iq}), \end{aligned} \tag{12}$$

where the expectation over future states has been replaced by a Monte Carlo estimator based on the trajectory data. In the limit of sampling an infinite number of trajectories ($m \to \infty$), the approximation given by Eq. (12) converges to the true value of g_i^{μ}.

A similar approximation can be constructed for the associated Bellman kernel by starting with Eq. (6):

$$\begin{aligned} \underline{\Psi}(i) &= \underline{\Phi}(i) - \alpha \sum_{j \in \mathcal{S}} P_{ij}^{\mu} \underline{\Phi}(j) \\ &= \underline{\Phi}(i) - \alpha \mathbb{E}_j \left[\underline{\Phi}(j) \right] \\ &\approx \underline{\Phi}(i) - \frac{\alpha}{m} \sum_{q=1}^{m} \underline{\Phi}(T_1^{iq}) \end{aligned} \tag{13}$$

Substituting Eq. (13) into Eq. (10) gives

$$\begin{aligned} \mathcal{K}(i, i') \approx \langle & \underline{\Phi}(i) - \frac{\alpha}{m} \sum_{q=1}^{m} \underline{\Phi}(T_1^{iq}), \\ & \underline{\Phi}(i') - \frac{\alpha}{m} \sum_{q'=1}^{m} \underline{\Phi}(T_1^{i'q'}) \rangle. \end{aligned}$$

Expanding this expression using linearity of the dot product gives

$$\begin{aligned} \mathcal{K}(i, i') \approx & \langle \underline{\Phi}(i), \underline{\Phi}(i') \rangle - \\ & \frac{\alpha}{m} \sum_{q=1}^{m} \left(\langle \underline{\Phi}(T_1^{iq}), \underline{\Phi}(i') \rangle + \langle \underline{\Phi}(T_1^{i'q}), \underline{\Phi}(i) \rangle \right) + \\ & \frac{\alpha^2}{m^2} \sum_{q=1}^{m} \sum_{q'=1}^{m} \langle \underline{\Phi}(T_1^{iq}, \underline{\Phi}(T_1^{i'q'} \rangle. \end{aligned}$$

Finally, substituting the definition of the kernel $k(i, i') = \langle \underline{\Phi}(i), \underline{\Phi}(i') \rangle$ gives

$$\begin{aligned} \mathcal{K}(i, i') \approx & k(i, i') - \frac{\alpha}{m} \sum_{q=1}^{m} \left(k(T_1^{iq}, i') + k(T_1^{i'q}, i) \rangle \right) \\ & \frac{\alpha^2}{m^2} \sum_{q=1}^{m} \sum_{q'=1}^{m} k(T_1^{iq}, T_1^{i'q'}). \end{aligned} \tag{14}$$

Again, in the limit of infinite sampling, Eq. (14) converges to $\mathcal{K}(i, i')$.

Finally, an approximation to $\widetilde{J}_{\mu}(i)$ [Eq. (11)] is needed:

$$\begin{aligned} \widetilde{J}_{\mu}(i) &= \sum_{i' \in \widetilde{\mathcal{S}}} \lambda_{i'} \left(k(i', i) - \alpha \sum_{j \in \mathcal{S}} P_{i'j}^{\mu} k(j, i) \right) \\ &= \sum_{i' \in \widetilde{\mathcal{S}}} \lambda_{i'} \left(k(i', i) - \alpha \mathbb{E}_j \left[k(j, i) \right] \right) \\ &\approx \sum_{i' \in \widetilde{\mathcal{S}}} \lambda_{i'} \left(k(i', i) - \frac{\alpha}{m} \sum_{q=1}^{m} k(T_1^{i'q}, i) \right). \end{aligned} \tag{15}$$

The procedure for carrying out model-free BRE can now be stated as follows:

1) Using the generative model of the MDP (or data collected from the real system), simulate m trajectories of length n starting from each of the sample states $i \in \widetilde{\mathcal{S}}$. Store this data in T_l^{iq}.
2) Solve the regression problem [Eq. (8)] using the stochastic approximations to g_i^{μ} and $\mathcal{K}(i, i')$ (given by Eqs. (12) and (14), respectively), and any kernel-based regression technique to find the coefficients $\{\lambda_{i'}\}_{i' \in \widetilde{\mathcal{S}}}$.
3) Use the coefficients found in step 2 to compute a stochastic approximation to the cost function $\widetilde{J}_{\mu}$, given by Eq. (15).
4) Perform a policy improvement step based on the cost function found in step 3, and return to step 1 to evaluate this new policy.

Computational Complexity

The overall complexity of running model-free is still dominated by building the kernel Gram matrix and solving the regression problem, just as in model-based BRE. Furthermore, the complexity of solving the regression problem is the same between both methods: $\mathcal{O}(n_s^3)$. However, in model-free BRE, computing an element of the associated Bellman kernel using Eq. (14) now requires $\mathcal{O}(m^2)$ operations (where m is the number of sampled trajectories), instead of $\mathcal{O}(\beta^2)$ (where β is the average branching factor of the MDP) as in the model-based case. In essence, in model-free BRE, the successor states for each sample state $i \in \widetilde{\mathcal{S}}$ (of which there are on average β) are approximated using data from m simulated trajectories. Thus, computing the full Gram matrix requires $\mathcal{O}(m^2 n_s^2)$ operations, and the total complexity of model-free BRE is of order

$$\mathcal{O}(m^2 n_s^2 + n_s^3).$$

Correctness of Model-Free BRE

The following theorem and lemma establish two important properties of the model-free BRE algorithm presented in this section.

Theorem: In the limit of sampling an infinite number of trajectories (i.e. $m \to \infty$), the cost function $\widetilde{J}_\mu(i)$ computed by model-free BRE is identical to the cost function computed by model-based BRE.

Proof: In order to approximate the quantities g_i^μ, $\mathcal{K}(i,i')$, and $\widetilde{J}_\mu(i)$ in the absence of model information, model-free BRE uses the stochastic approximators given by Eqs. (12), (14), and (15), respectively. Examining these equations, note that in each, the state trajectory data T_l^{iq} is used to form an empirical distribution

$$\hat{P}_{ij}^\mu = \frac{1}{m}\sum_{q=1}^{m} \delta(T_1^{iq}, j),$$

where $\delta(i,j)$ is the Kronecker delta function. This distribution is used as a substitute for the true distribution P_{ij}^μ in computing g_i^μ, $\mathcal{K}(i,i')$, and $\widetilde{J}_\mu(i)$. Since by assumption the individual trajectories are independent, the random variables $\{T_1^{iq} | q = 1, \ldots, m\}$ are independent and identically distributed. Therefore, the law of large numbers states that the empirical distribution converges to the true distribution in the limit of an infinite number of samples:

$$\lim_{m\to\infty} \hat{P}_{ij}^\mu = P_{ij}^\mu.$$

Therefore, as $m \to \infty$, Eqs. (12), (14), and (15) converge to the true values g_i^μ, $\mathcal{K}(i,i')$, and $\widetilde{J}_\mu(i)$. In particular, the cost function $\widetilde{J}_\mu(i)$ computed by model-free BRE converges to the cost function computed by model-based BRE as $m \to \infty$, as desired. ■

Using results shown in [1], [2], which prove that model-based BRE computes a cost function $\widetilde{J}_\mu(i)$ whose Bellman residuals are exactly zero at the sample states $\widetilde{\mathcal{S}}$, we immediately have the following lemma:

Lemma: In the limit $m \to \infty$, model-free BRE computes a cost function $\widetilde{J}_\mu(i)$ whose Bellman residuals are exactly zero at the sample states $\widetilde{\mathcal{S}}$.

Proof: The theorem showed that in the limit $m \to \infty$, model-free BRE yields the same cost function $\widetilde{J}_\mu(i)$ as model-based BRE. Therefore, applying the results from [1], [2], it immediately follows that the Bellman residuals $\widetilde{J}_\mu(i)$ are zero at the sample states. ■

IV. Results

A further series of tests were carried out to compare the performance of the BRE algorithms presented in this paper to LSPI [3], a well-known approximate dynamic programming algorithm. LSPI uses a linear combination of basis functions to represent approximate Q-values of the MDP, and learns a weighting of these basis functions using simulated trajectory data. In learning the weights, LSPI can take advantage of system model information if it is available, or can be run in a purely model-free setting if not. Thus, our tests compared the performance of four different algorithms: model-based BRE, model-free BRE, model-based LSPI, and model-free LSPI. The LSPI implementation used for these tests is the one provided by the authors in [3].

The benchmark problem used was the "chain-walk" problem [3], [17], which has a one-dimensional state space (we used a total of 50 states in these tests) and two possible actions ("move left" and "move right") in each state. In order to make the comparisons between BRE and LSPI as similar as possible, the same cost representation was used in both algorithms. More precisely, in LSPI, five radial basis functions (with standard deviation $\sigma = 12$), with centers at $x = 1, 11, 21, 31, 41$, were used; while in BRE, the same radial basis kernel (with the same σ) was used and the sample states were taken as $\widetilde{\mathcal{S}} = \{1, 11, 21, 31, 41\}$. This ensures that neither algorithm gains an unfair advantage by being provided with a better set of basis functions.

The algorithms were compared on two different performance metrics: quality of the approximate policy produced by each algorithm (expressed as a percentage of states in which the approximate policy matches the optimal policy), and running time of the algorithm (measured in the total number of elementary operations required, such as additions and multiplications). The policies produced LSPI and the model-free variant of BRE depend on the amount of simulated data provided to the algorithm, so each of these algorithms was run with different amounts of simulated data to investigate how the amount of data impacts the quality of the resulting policy. Furthermore, since the simulated data is randomly generated according to the generative model, each algorithm was run multiple times to examine the effect of this random simulation noise on the algorithm performance.

The results of the comparison tests are shown in Fig. 2. Model-based BRE, shown as the filled blue diamond, finds the optimal policy. Furthermore, its running time is faster than any other algorithm in the test by at least one order of magnitude. In addition, it is the only algorithm that is free from simulation noise, and it therefore consistently finds the optimal policy every time it is run, unlike the other algorithms which may yield different results over different runs. Thus, if a system model is available, model-based BRE has a clear advantage over the other algorithms in the chain-walk problem.

LSPI also found the optimal policy in a number of the tests, confirming similar results that were reported in [3]. However, the amount of simulation data required to consistently find a near-optimal policy was large (between 5,000 and 10,000 samples), leading to long run times. Indeed, for any of the LSPI variants that consistently found policies within 10% of optimal, all were between two and four orders of magnitude slower than model-based BRE. As the amount of simulation data is decreased in an attempt to reduce the solution time of LSPI, the quality of the produced policies becomes lower on-average and also more inconsistent across runs. For simulated data sets of less than about 1,000, the policy quality approaches (and sometimes drops below) 50% of optimal, indicating performance equivalent to (or worse than) simply guessing one of the two actions randomly. Al-

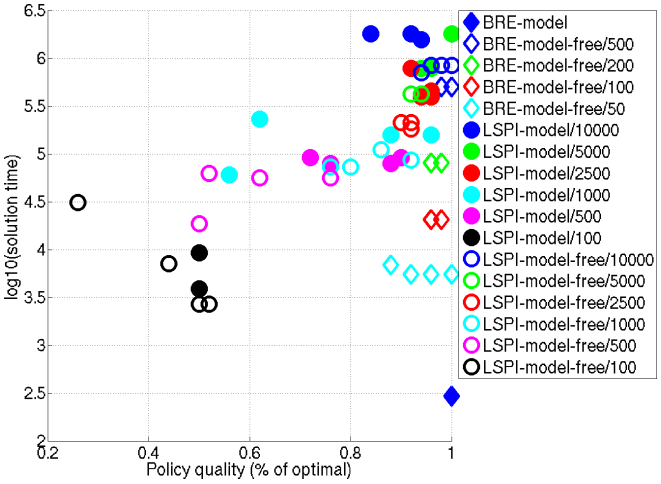

Fig. 2: Comparison of BRE vs. LSPI for the "chain-walk" problem. Numbers after the algorithm names denote the amount of simulation data used to train the algorithm. Note that model-based BRE does not require simulation data.

lowing LSPI access to the true system model does appear to improve the performance of the algorithm slightly; in Fig. 2, model-based LSPI generally yields a higher quality policy than model-free LSPI for a given data set size (although there are several exceptions to this). However, the results indicate the model-based BRE is significantly more efficient than model-based LSPI at exploiting knowledge of the system model to reduce the computation time needed to find a good policy.

Finally, examining the results for model-free BRE, the figure shows that this algorithm yields consistently good policies, even when a small amount of simulation data is used. As the amount of simulation data is reduced, the variability of the quality of policies produced increases and the average quality decreases, as would be expected. However, both of these effects are significantly smaller than in the LSPI case. Indeed, the worst policy produced by model-free BRE is still within 12% of optimal, while using only 50 simulation data. In contrast, LSPI exhibited a much greater variability and much lower average policy quality when the amount of simulation data was reduced.

V. Conclusion

This paper has presented a model-free variant of the BRE approach to approximate dynamic programming. Instead of relying on knowledge of the system model, the model-free variant uses trajectory simulations or data from the real system to build stochastic approximations to the cost values g_i^μ and kernel Gram matrix $\mathbb{K}$ needed to carry out BRE. It is straightforward to show that in the limit of carrying out an infinite number of simulations, this approach reduces to the model-based BRE approach, and thus enjoys the same theoretical properties of that approach (including the important fact that the Bellman residuals are identically zero at the sample states $\widetilde{\mathcal{S}}$.

Furthermore, experimental comparison of BRE against another well-known approach, LSPI, indicates that while both approaches find near-optimal policies, both model-based and model-free BRE appears to have several advantages over LSPI in the benchmark problem we tested. In particular, model-based BRE appears to be able to efficiently exploit knowledge of the system model to find the policy significantly faster than LSPI. In addition, model-based BRE is free of simulation noise, eliminating the problem of inconsistent results across different runs of the algorithm. Even when model information is not available, model-free BRE still finds near-optimal policies more consistently and more quickly than LSPI.

Acknowledgments

Research supported by the Boeing Company, Phantom Works, Seattle; AFOSR grant FA9550-08-1-0086; and the Hertz Foundation.

References

[1] B. Bethke and J. How, "Approximate dynamic programming using Bellman residual elimination and Gaussian process regression," in *Proceedings of the American Control Conference*, St. Louis, MO, 2009.

[2] B. Bethke, J. How, and A. Ozdaglar, "Approximate dynamic programming using support vector regression," in *Proceedings of the 2008 IEEE Conference on Decision and Control*, Cancun, Mexico, 2008.

[3] M. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[4] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2007.

[5] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[6] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[7] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[8] Y. Engel, "Algorithms and representations for reinforcement learning," Ph.D. dissertation, Hebrew University, 2005.

[9] M. Deisenroth, J. Peters, and C. Rasmussen, "Approximate dynamic programming with Gaussian processes," in *Proceedings of the American Control Conference*, 2008.

[10] J. Si, A. Barto, W. Powell, and D. Wunsch, *Learning and Approximate Dynamic Programming*. NY: IEEE Press, 2004. [Online]. Available: http://citeseer.ist.psu.edu/651143.html

[11] B. Bethke, J. How, and J. Vian, "Multi-UAV persistent surveillance with communication constraints and health management," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Chicago, IL, August 2009.

[12] P. Schweitzer and A. Seidman, "Generalized polynomial approximation in Markovian decision processes," *Journal of mathematical analysis and applications*, vol. 110, pp. 568–582, 1985.

[13] L. C. Baird, "Residual algorithms: Reinforcement learning with function approximation." in *ICML*, 1995, pp. 30–37.

[14] R. Munos and C. Szepesvári, "Finite-time bounds for fitted value iteration," *Journal of Machine Learning Research*, vol. 1, pp. 815–857, 2008.

[15] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, pp. 337–404, 1950.

[16] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[17] D. Koller and R. Parr, "Policy iteration for factored MDPs," in *UAI*, C. Boutilier and M. Goldszmidt, Eds. Morgan Kaufmann, 2000, pp. 326–334.