

---

# Kernel-Based Models for Reinforcement Learning

---

Nicholas K. Jong

Peter Stone

Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712 USA

NKJ@CS.UTEXAS.EDU

PSTONE@CS.UTEXAS.EDU

## Abstract

Model-based approaches to reinforcement learning exhibit low sample complexity while learning nearly optimal policies, but they are generally restricted to finite domains. Meanwhile, function approximation addresses continuous state spaces but typically weakens convergence guarantees. In this work, we develop a new algorithm that combines the strengths of Kernel-Based Reinforcement Learning, which features instance-based state representation and kernel-based function approximation, and Prioritized Sweeping, which features model-based exploration. The resulting algorithm, Kernel-Based Prioritized Sweeping, empirically converges to good policies in continuous domains with relatively small amounts of data.

## 1. Introduction

Research into reinforcement learning (RL) (Sutton & Barto, 1998) addresses an extraordinarily general problem: how to learn good behavior policies in arbitrary unknown environments. Elegant and computationally efficient canonical algorithms, such as Q-learning (Watkins, 1989), belie the difficulty of this problem by promising asymptotic convergence to optimal behavior for any finite domain. Practical applications challenge RL techniques in at least two ways. First, the cost of obtaining data implies a need for algorithms with low sample complexity. Second, continuous state spaces require generalization methods that preserve convergence to good policies.

Model-based approaches address the sample complexity issue by directly estimating the effect of each action at each state. This approach facilitates the efficient reuse of data, and reasoning about the uncer-

tainty in the model allows these algorithms to balance exploration and exploitation. Algorithms such as Prioritized Sweeping<sup>1</sup> exhibit good sample complexity for finite problems (Moore & Atkeson, 1993). Another model-based algorithm, E<sup>3</sup>, provides the first polynomial-time convergence guarantees (Kearns & Singh, 1998). However, these approaches do not generalize directly to continuous problems with stochastic dynamics, due to the difficulty in representing and reasoning with continuous distributions over possible transitions.

Current RL algorithms for continuous state spaces typically rely on function approximation to generalize the value function directly, without estimating a model. Kernel-based reinforcement learning (KBRL) computes a value function offline by generalizing value-function updates from a given sample of transitions over an instance-based representation (Ornstone & Sen, 2002). KBRL is noteworthy for its theoretical guarantee of convergence to the optimal value function as its sample size increases, under appropriate assumptions, but it does not answer the exploration question of how to efficiently gather the data online.

The primary contribution of this paper is a novel algorithm that bridges the gap between function approximation, which handles continuous state spaces, and the model-based approach, which handles intelligent exploration. It also contributes a new perspective on KBRL that emphasizes its connection to model-based RL. This perspective permits the combination of the function approximation of KBRL with the exploration mechanism of Prioritized Sweeping. The result is a practical algorithm, Kernel-Based Prioritized Sweeping (KBPS), which solves continuous domains with empirically small amounts of data.

---

<sup>1</sup>Although this name has come to be associated with a particular method for updating a value function, here we denote the entire RL algorithm originally presented under that name.

## 2. Preliminaries

This section describes background and prior work. First, we review the formalism that underlies value-based approaches to RL. Then we review Prioritized Sweeping, a relatively straightforward model-based approach to RL that learns efficiently in finite state spaces. Finally, we summarize KBRL, an instance-based algorithm for computing value functions over continuous state spaces.

### 2.1. Markov Decision Processes

A RL task is defined by a Markov decision problem (MDP), which is a four tuple  $\langle S, A, T, R \rangle$  consisting of a finite state space  $S$ , a finite action space  $A$ , a transition function  $T : S \times A \times S \rightarrow \mathbb{R}$ , and a reward function  $R : S \times A \rightarrow \mathbb{R}$  (Puterman, 1994). From a given state  $s \in S$ , a given action  $a \in A$  produces an expected reward of  $R(s, a)$  and transitions to another state  $s' \in S$  with probability  $T(s, a, s')$ .

Given an MDP, planning algorithms compute a policy  $\pi : S \rightarrow A$  that maximizes cumulative rewards. Classical approaches employ dynamic programming to compute the value function  $V : S \rightarrow \mathbb{R}$ , where  $V(s)$  is the maximum expected cumulative reward possible from  $s$ . An optimal policy  $\pi$  satisfies  $\pi(s) = \operatorname{argmax}_a Q(s, a)$ , where  $Q : S \times A \rightarrow \mathbb{R}$  is defined by  $Q(s, a) = R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s')$ . The discount factor  $\gamma \in [0, 1]$  is sometimes necessary to ensure that the value function does not diverge.

### 2.2. Model-Based RL

RL algorithms compute policies without the benefit of knowing the transition and reward functions  $T$  and  $R$ . Model-free algorithms compute a value function directly from experience data, but model-based algorithms first estimate the underlying MDP from the experience data. Standard MDP planning techniques, such as value iteration (Puterman, 1994), can then compute an optimal policy from the estimated model.

For finite domains, learning the model online is straightforward. It suffices to maintain for each state  $s$  and each action  $a$ : the total one-step reward  $r^{s,a}$  earned by executing  $a$  in  $s$ , the total number of times  $n^{s,a}$  that  $a$  has been executed in  $s$ , and for each other state  $s'$  the total number of times  $n^{s,a,s'}$  that executing  $a$  in  $s$  transitioned to  $s'$ . Then, at any point in time, the maximum-likelihood estimates of the transition and reward functions are  $\hat{T}(s, a, s') = n^{s,a,s'} / n^{s,a}$  and  $\hat{R}(s, a) = r^{s,a} / n^{s,a}$ .

One practical concern for online model-based RL is

how to plan efficiently given the changing model. Moore and Atkeson (1993) showed how to update the estimated value function  $\hat{V}$  incrementally given an incremental update to the model. Whereas typical value iteration updates the value of every state during each iteration, Prioritized Sweeping maintains a priority queue of states. Whenever  $\hat{V}(s')$  changes by an amount  $\delta$ , the algorithm adds each state  $s$  to the priority queue with priority  $\max_a \hat{T}(s, a, s') \delta$ . Each time the agent takes an action, the algorithm first updates one state-action pair in the model, then updates the value of that state-action pair, and finally uses the available computational time to propagate changes throughout the value function by recomputing  $\hat{V}$  for the first states in the priority queue.

The Prioritized Sweeping algorithm also includes a simple model-based approach to the exploration problem. For a given state  $s$  and action  $a$ , it only uses the maximum-likelihood estimates  $\hat{T}(s, a, \cdot)$  and  $\hat{R}(s, a)$  if  $n^{s,a}$  exceeds some threshold  $n_{\text{known}}$ . Below this threshold, the algorithm assumes that not enough data exists to accurately estimate the outcome of executing  $a$  in  $s$ , so it optimistically assumes a deterministic transition to a fictional absorbing state  $s_{\text{final}}$  with constant value  $V(s_{\text{final}}) = \frac{r_{\text{max}}}{1-\gamma}$ , where  $r_{\text{max}}$  is some upper bound on the one-step reward. The algorithm thus computes an optimistic value function  $V^{\text{opt}}$  from an optimistic model  $\langle S \cup \{s_{\text{final}}\}, A, \hat{T}^{\text{opt}}, \hat{R}^{\text{opt}} \rangle$ , where for all  $s \in S$ ,  $a \in A$ , and  $s' \in S$ :

$$\hat{T}^{\text{opt}}(s, a, s') = \begin{cases} \hat{T}(s, a, s'), & \text{if } n^{s,a} \geq n_{\text{known}} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$\hat{R}^{\text{opt}}(s, a) = \begin{cases} \hat{R}(s, a), & \text{if } n^{s,a} \geq n_{\text{known}} \\ r_{\text{max}}, & \text{otherwise.} \end{cases} \quad (2)$$

We define  $\hat{T}^{\text{opt}}(s, a, s')$  in the appropriate way if  $s = s_{\text{final}}$  or  $s' = s_{\text{final}}$ . Thus, if the agent is in a state  $s$  with an insufficiently explored action  $a$ , then  $\hat{Q}^{\text{opt}}(s, a) = V_{\text{max}}$  and the agent will choose  $a$  (or another exploratory action if more than one exists). Furthermore, since the agent propagates these optimistic values to the rest of the state space, it navigates to unknown regions so long as the optimistic cost of reaching the exploratory states is small compared to rewards available in the known regions.

This directed form of exploration helps to minimize the amount of experience that Prioritized Sweeping requires to learn. What data the algorithm does collect it tabulates into a model, which it uses repeatedly to update the value function. However, this data-efficient approach does not generalize directly to problems with continuous state spaces and stochastic dynamics, given the lack of an efficient way to estimate the transition

function. In domains such as robotics, the assumption of deterministic dynamics permits the use of regression to learn the model (Atkeson et al., 1997), but this method only allows reasoning about the average outcome, not the set of possible outcomes.

### 2.3. Kernel-Based RL

In contrast, KBRL is an approach that computes continuous value functions directly from a set of historical outcomes (Ormoneit & Sen, 2002). Intuitively, KBRL approximates the outcome of an action  $a$  from a given state  $s$  as the average of previous outcomes of that action, weighted by a function of the distances between  $s$  and the previous initial states. Let  $s_1, \dots, s_n$  be a set of states sampled from a continuous state space  $S$  that has a distance metric  $d$ . Suppose that for each state  $s_i$  we have an associated transition: an executed action  $a_i$ , observed reward  $r_i$ , and observed successor state  $s'_i$ . Then KBRL defines approximate Bellman equations for all  $s \in S$  and all  $a \in A$  as follows:

$$\hat{Q}(s, a) = \frac{1}{Z^{s,a}} \sum_{i|a_i=a} \phi\left(\frac{d(s_i, s)}{b}\right) [r_i + \gamma \hat{V}(s'_i)], \quad (3)$$

where  $\hat{V}(s) = \max_a \hat{Q}(s, a)$  as usual,  $b$  is a bandwidth parameter that scales the distance function,  $\phi$  is a non-negative function that determines the relative weight of each transition, and  $Z^{s,a} = \sum_{j|a_j=a} \phi\left(\frac{d(s_j, s)}{b}\right)$  normalizes the weights. We use Gaussian kernels to compute these weights, so  $\phi(x) = e^{-x^2}$ . The bandwidth parameter  $b$  thus corresponds to the standard deviation of each Gaussian.

Let  $D = \{s'_i\}$  be the set of observed successor states. Although  $\hat{V}$  is a function over a continuous space  $S$ , we can perform value iteration feasibly by storing and updating  $\hat{V}(D)$ , since Equation 3 only evaluates  $\hat{V}$  on the finite subset  $D$ . Assuming that the true transition and reward functions are suitably well behaved and the transitions for each action are sampled uniformly, the unique fixed point of this value iteration converges in probability to the true value function as the sample size grows, although the bandwidth  $b$  must decrease at a suitable rate to balance the variance and bias of  $\hat{V}$  (Ormoneit & Sen, 2002).

With its instance-based value-function representation and kernel-based generalization, KBRL offers accurate value-function approximation for continuous RL problems. However, Ormoneit and Sen (2002) construe KBRL principally as an offline algorithm and do not address the question of exploration versus exploitation. In the next section, we will recast KBRL as a

method for approximating a continuous domain with finite model, permitting the use of model-based exploration techniques.

## 3. Kernel-Based Models for RL

Here we combine the strengths of the two RL approaches described in Sections 2.2 and 2.3 to produce an online, model-based algorithm that accurately handles stochastic environments with continuous state spaces. For our purposes, we explicitly cast KBRL as a means to approximate an unknown continuous MDP  $M$  with a finite MDP  $\tilde{M}$  defined over sampled transitions. As before, let  $D$  be the set of sampled successor states. We define a transition function  $\tilde{T} : S \times A \times D \rightarrow \mathbb{R}$  and reward function  $\tilde{R} : S \times A \rightarrow \mathbb{R}$  as follows for all  $s \in S$ ,  $a \in A$ , and  $s'_i \in D$ :<sup>2</sup>

$$\tilde{T}(s, a, s'_i) = \begin{cases} \frac{1}{Z^{s,a}} \phi\left(\frac{d(s_i, s)}{b}\right) & \text{if } a_i = a \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$\tilde{R}(s, a) = \frac{1}{Z^{s,a}} \sum_{i|a_i=a} \phi\left(\frac{d(s_i, s)}{b}\right) r_i. \quad (5)$$

Since  $D \subset S$ , we observe that  $\tilde{M} = \langle D, A, \tilde{T}, \tilde{R} \rangle$  is a well defined finite MDP. Note that for all  $s \in D$  the definition in Equation 3 of the approximate value function  $\hat{Q}$  for the continuous MDP  $M$  corresponds exactly to the definition of the true value function  $\tilde{Q}$  for the finite model  $\tilde{M}$ :  $\hat{Q}(s, a) = \tilde{R}(s, a) + \gamma \sum_{s'_i} \tilde{T}(s, a, s'_i) \tilde{V}(s)$ . This perspective yields a much simpler proof that KBRL converges to a unique fixed point than the one Ormoneit and Sen offered, merely by construing the algorithm as computing the exact value function for an approximate finite MDP instead of computing an approximate value function for an unknown continuous MDP. Conversely, the theoretical results of Ormoneit and Sen (2002) imply that, as the sample size grows, for every  $s \in D$ , the value  $\tilde{V}(s)$  in the approximate finite MDP converges to the true value  $V(s)$  in the continuous MDP. Furthermore, we can accurately compute  $\hat{Q}(s, a)$  for any state in the original continuous state space by simply using Equation 3, substituting  $\tilde{V}$  for  $\hat{V}$  in the right-hand side, since these two value functions agree on  $D$ .

### 3.1. Online KBRL

The ability to import existing model-based techniques is a key practical benefit of treating KBRL as an algorithm that transforms data into an approximate fi-

<sup>2</sup>For the sake of convenience, we abuse notation by quantifying over  $s'_i$  instead of  $s'$ . Note that for any  $s' \in D$  we may choose  $i$  such that  $s'_i = s'$ .

nite MDP, instead of into an approximate value function. Many model-based approaches depend principally on tabulated data in the form of the quantities  $n^{s,a}$ ,  $n^{s,a,s'}$ , and  $r^{s,a}$  for all  $s$ ,  $a$ , and  $s'$ , defined in Section 2.2. For a set of sample transitions from a continuous MDP  $M$ , we can define “pseudodata” for the induced MDP  $\tilde{M}$  as follows:

$$\tilde{n}^{s,a,s'} = \begin{cases} \phi\left(\frac{d(s_i,s)}{b}\right) & \text{if } a_i = a \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$\tilde{n}^{s,a} = Z^{s,a} = \sum_i \tilde{n}^{s,a,s'_i} \quad (7)$$

$$\tilde{r}^{s,a} = \sum_{i|a_i=a} \phi\left(\frac{d(s_i,s)}{b}\right) r_i. \quad (8)$$

Computing the maximum-likelihood transition and reward functions from this pseudodata yields the estimates given in Equations 4 and 5. This identity suggests a general method for adapting model-based algorithms developed for finite domains to the continuous case. We simply replace the standard model update given a new transition  $(s_t, a_t, r_t, s_{t+1})$  with the following procedure:

---

**Algorithm 1** GROWMODEL( $s_t, a_t, r_t, s_{t+1}$ )
 

---

- 1: Add  $s_{t+1}$  to the set of states  $D$
  - 2: **for all**  $a$  **do** {Define the model at the new state}
  - 3:   **for all**  $s'_i$  **do**
  - 4:     Initialize  $\tilde{n}^{s_{t+1},a,s'_i}$  according to Equation 6
  - 5:     Initialize  $\tilde{n}^{s_{t+1},a}$  according to Equation 7
  - 6:     Initialize  $\tilde{r}^{s_{t+1},a}$  according to Equation 8
  - 7:   **end for**
  - 8: **end for**
  - 9: **for all**  $s'_i \neq s_{t+1}$  **do** {Update other states}
  - 10:   Initialize  $\tilde{n}^{s'_i,a_t,s_{t+1}}$  according to Equation 6
  - 11:   Add  $\tilde{n}^{s'_i,a_t,s_{t+1}}$  to  $\tilde{n}^{s'_i,a_t}$
  - 12:   Add  $\tilde{n}^{s'_i,a_t,s_{t+1}} r$  to  $\tilde{r}^{s'_i,a_t}$
  - 13: **end for**
- 

Applying this approach to Prioritized Sweeping yields a new algorithm:

---

**Algorithm 2** Kernel-Based Prioritized Sweeping
 

---

- 1: **repeat**
  - 2:   In state  $s$ , execute  $a$ , receive  $r$ , and observe  $s'$
  - 3:    $a' \leftarrow \operatorname{argmax}_a \hat{Q}^{\text{opt}}(s', a)$
  - 4:   GROWMODEL( $s, a, r, s'$ )
  - 5:   Compute  $\tilde{V}^{\text{opt}}(s)$
  - 6:   Propagate changes to  $\tilde{V}^{\text{opt}}$
  - 7:    $s \leftarrow s'$
  - 8:    $a \leftarrow a'$
  - 9: **until**  $s'$  is terminal
- 

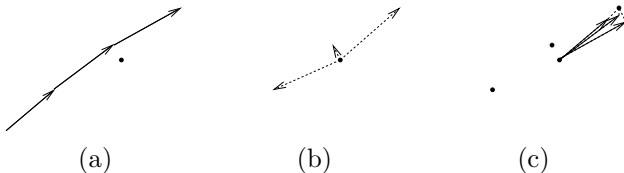
Note that in line 3 we compute the value function at a new state  $s_{t+1}$  *before* adding  $s_{t+1}$  to the model, using an optimistic version of Equation 3. This slight modification to the order of steps in Prioritized Sweeping allows us to initialize  $\tilde{V}(s_{t+1})$  to a reasonable value. In the next iteration of the main loop, we recompute  $\tilde{V}(s_{t+1})$  in line 5, after  $s_{t+1}$  as been added to the model. We use prioritized value updates, as described in Moore and Atkeson (1993), to propagate the change in value due to adding the transition. Provided that we allow enough value updates to converge, the value function after each iteration of the loop will be equal to the value function that KBRL would compute offline given the same data, modified only to include Prioritized Sweeping’s optimistic exploration strategy. We refer to this algorithm as Kernel-Based Prioritized Sweeping (KBPS).

### 3.2. Relative Transition Models

Early experiments with KBPS revealed a tendency to learn poor policies. In practice, the bandwidth parameter  $b$  must be large enough to prevent KBPS from taking an enormous number of exploratory actions, to visit every neighborhood of the state space. However, a relatively high amount of generalization introduces bias into the learned model. This section examines a primary source of bias for many typical application domains and proposes a modification to the KBRL approximation scheme that significantly improves the accuracy of KBPS in these domains.

Consider computing the transition function for the state shown in Figure 1a, which shows three nearby sample transitions for one of the actions. Our approximate model predicts that the successor state from the given point will be one of these three previously observed successor states, as shown in Figure 1b. Although the action in question consistently shifted the state in the same general direction, to the upper right, the approximate model assigns substantial probability to moving in the opposite direction. With sufficiently biased data, this unlikely outcome may even become the expected outcome according to the approximate model. The problem is that the generalization breadth of the kernel is large relative to the magnitude of the action’s effect. As a result, the location of a predicted successor states depends as much on the historical location of the initial state as on the action itself.

To ameliorate the problem, we propose a modification to the kernelized model. Given a state  $s$ , this model uses a sample transition from  $s_i$  to  $s'_i$  not to predict an absolute transition to  $s'_i$  but to predict a relative transition to  $s + (s'_i - s_i)$ , as illustrated in Figure 1c. How-



**Figure 1:** (a) A query point in some state space with three nearby sample transitions. (b) Three possible transitions predicted by KBRL. (c) Transitions predicted by a relative transition model.

ever, this predicted successor state is unlikely to reside in  $D$ , the set of previously visited successor states, making it difficult to estimate the value of  $s$ , which depends on the values of its successors. Hence, although this approach better captures the true dynamics of the system, it prevents efficient planning, which relies on the transition model being closed under  $D$ . We rectify this problem by using a second kernel to approximate a transition to  $s + (s'_i - s_i)$  as a weighted combination of transitions to the states in  $D$ . To this end, we define the pseudodata for a particular model transition as follows:

$$\tilde{n}^{s,a,s'} = \sum_{i|a_i=a} \phi\left(\frac{d(s_i, s)}{b}\right) \frac{\phi\left(\frac{d(s', s+(s'_i-s_i))}{h}\right)}{w^{s,i}}, \quad (9)$$

where  $h$  is another bandwidth parameter and  $w^{s,i} = \sum_{s' \in D} \phi\left(\frac{d(s', s+(s'_i-s_i))}{h}\right)$  normalizes the weights. Note that the first application of the kernel  $\phi$  weights the contribution of each transition towards the transition model for a given state, with bandwidth parameter  $b$ , and the second application, normalized by  $w^{s,i}$ , weights the contribution of each successor state towards the approximation of each transition, with bandwidth parameter  $h$ .

This formulation reveals a second source of uncertainty in the approximate model. The first source, measured by  $n^{s,a}$ , arises from having too few relevant sample transitions to estimate the change that an action causes. The second source, measured by  $w^{s,i}$ , arises from having too few nearby states to approximate the state resulting from a given change. In KBPS, we can behave optimistically with regard to the second source using an analogous exploration mechanism. In particular, we define optimistic pseudodata

$$\tilde{n}_{opt}^{s,a,s'} = \sum_{\substack{i|a_i=a \\ w^{s,i} \geq w_{\text{known}}}} \phi\left(\frac{d(s_i, s)}{b}\right) \frac{\phi\left(\frac{d(s', s+(s'_i-s_i))}{h}\right)}{w^{s,i}}, \quad (10)$$

where  $w_{\text{known}}$  controls the number of nearby model states required to approximate a transition outcome.

Note that unlike the absolute transition model, this relative transition model may lead to transition functions that assign some intermediate probability to reaching the fictional absorbing state  $s_{\text{final}}$  from a given state  $s$  and action  $a$ . In particular,

$$\tilde{n}_{opt}^{s,a,s_{\text{final}}} = \sum_{i|a_i=a \wedge w^{s,i} < w_{\text{known}}} \phi\left(\frac{d(s_i, s)}{b}\right). \quad (11)$$

The algorithm that results from substituting Equations 10 and 11 for Equation 6 in Lines 4 and 10 of Algorithm 1 explores either if an action hasn't been attempted near a certain state or if the model predicts that an action may transition to an unvisited region of the state space. The introduction of approximation for the outcomes of individual transitions also permits more freedom in the choice of  $D$ , the states used to represent the continuous state space.

## 4. Experimental Results

KBPS is designed for sample efficiency via a combination of model-based exploration, sample-based state representation, and kernel-based generalization. This section describes our experiments, which demonstrate that KBPS converges more rapidly to near-optimal policies, compared to several other recent RL algorithms evaluated on some benchmark problems with continuous state spaces.

### 4.1. Implementation Details

A primary practical concern for an instance-based algorithm such as KBPS is computational complexity. In fact, the casting of KBRL as a transformation into an approximate model arose in part from caching considerations for a more straightforward implementation. Once the finite model has been built, fast tabular methods can reference it to update the value function. However, the GROWMODEL procedure, which sets the transition probabilities to and from a new state, requires running time linear in the size of  $D$ , which is equal to the number of transitions already experienced.

Our implementation achieves a substantial reduction in the constant factor of GROWMODEL's  $O(D)$  running time by observing that the procedure only changes the model appreciably in a local region of the state space. It modifies the Gaussian kernel to return 0 instead of values smaller than 0.01. Thus the addition of a new transition from  $s$  only affects the model at those states within distance  $b\sqrt{-\log 0.01} = 2.146b$  from  $s$ , in the standard transition model. For the relative transition model, this sphere of influence increases in radius to  $2.146(b+h)$ . Our implementation

also prunes any kernel weights with normalized values smaller than 0.01 (and renormalizes the remaining weights), bounding the number of sample transitions (or successor states for the relative action model) used in each approximation to 100. Note that this pruning does not bias the approximation, since the probability of an individual successor in the model is a function of a sample transition’s distance from the model state, which is independent of the actual likelihood of the transition in the true MDP.

With these two pruning mechanisms,  $O(\log |D|)$  updates are possible by using a data structure such as a  $k$ -d tree to find the 100 nearest neighbors (for each action) of each model state. However, the following sections describe results obtained with a simple linear-time binning approach that searched for neighbors in adjacent bins. Due to memory and time constraints, our current implementation stops adding new data after sampling 10000 transitions.

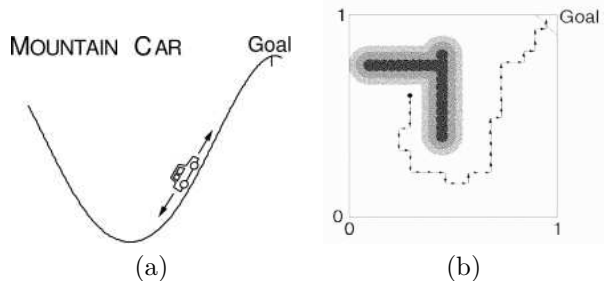
## 4.2. Benchmark Performance

This section compares the performance of KBPS to algorithms submitted to the RL benchmarking workshop held at NIPS 2005 (Dutech et al., 2005). This event invited researchers to implement algorithms in a common interface for online RL. Participants computed their results locally, but direct comparisons are possible due to the standardized environment code, which presents the same sequence of initial states to each algorithm. Sections 4.2.1 and 4.2.2 examine two of the benchmark domains and give the KBPS parameters used to solve them. Section 4.2.3 evaluates the performance of KBPS against selected algorithms.

### 4.2.1. MOUNTAIN CAR

In the Mountain Car simulation (Sutton & Barto, 1998), an underpowered car must escape a valley (Figure 2a) by backing up the left slope to build sufficient energy to reach the top of the right slope. The agent has two state variables, horizontal position  $x$  and horizontal velocity  $v$ . The three available actions are **reverse**, **neutral**, and **forward**, which add  $-0.001$ ,  $0$ , and  $0.001$  to  $v$ , respectively. In addition, gravity adds  $-0.0025 \cos(3x)$  to  $v$  at each time step. The agent receives a reward of  $-1$  for each time step before reaching the goal state. Episodes begin in a uniformly random initial position  $x$  and with  $v = 0$ , and they last for at most 300 time steps. The only domain knowledge available is the maximum one-step reward  $r_{\max} = 0$  and the minimum and maximum values of each state variable:  $-1.2$  and  $0.5$  for  $x$  and  $-0.07$  and  $0.07$  for  $v$ .

KBPS scaled both state variables to  $[0, 1]$ . The band-



**Figure 2:** Two of the domains from the NIPS benchmarking workshop: (a) Mountain Car and (b) Puddle World.

width parameters were  $b = 0.03$  to generalize transitions and  $h = 0.01$  to generalize successor states. Since Mountain Car is deterministic, the exploration thresholds were  $n_{\text{known}} = 1$  and  $w_{\text{known}} = 1$ . To compute the value function, KBPS applied at most 1000 updates with minimum priority 0.01 after each transition.

### 4.2.2. PUDDLE WORLD

The Puddle World (Sutton, 1996) is a continuous grid world with the goal in the upper-right corner and two oval puddles (Figure 2b). The two state variables are the  $x$  and  $y$  coordinates, and the four actions correspond to the four cardinal directions. Each action moves the agent 0.05 in the indicated direction, with Gaussian noise added to each dimension with  $\sigma = 0.01$ . The agent receives a  $-1$  reward for each action outside of the two puddles, which extend with radius 0.1 from two line segments, one from  $(0.1, 0.75)$  to  $(0.45, 0.75)$  and the other from  $(0.45, 0.4)$  to  $(0.45, 0.8)$ . Being in a puddle incurs a negative reward equal to 400 times the distance inside the puddle. The goal region satisfies  $x + y \geq 0.95 + 0.95$ .

For this domain, KBPS used bandwidth parameters  $b = 0.05$  and  $h = 0.02$ . Although Puddle World is stochastic, thresholds  $n_{\text{known}} = 1$  and  $w_{\text{known}} = 1$  continued to suffice. KBPS used at most 1000 updates after each transition, with minimum priority 0.01.

### 4.2.3. BENCHMARK RESULTS

Figures 3 and 4 compare the performance of KBPS to three selected algorithms. (Each point is the average of fifty sequential episodes, as reported to the NIPS workshop.) These three algorithms, implemented and parameterized by other researchers, were among the most competitive submitted. One is a model-based approach applied to a fixed discretization of the state space. This algorithm employed the same exploration mechanism as Prioritized Sweeping, but it lacked the instance-based representation and kernel-based generalization of KBPS. Least Squares Policy Iteration

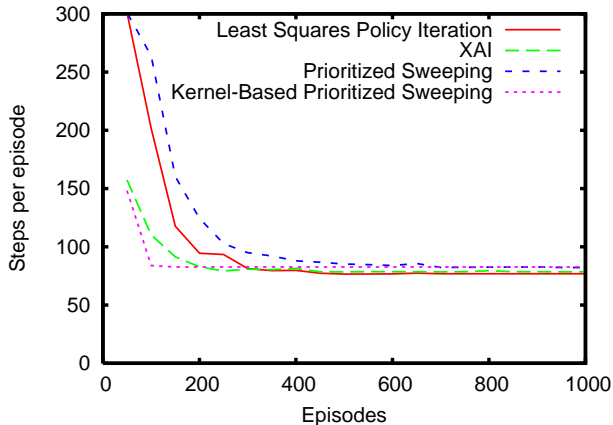


Figure 3: Learning curves for Mountain Car

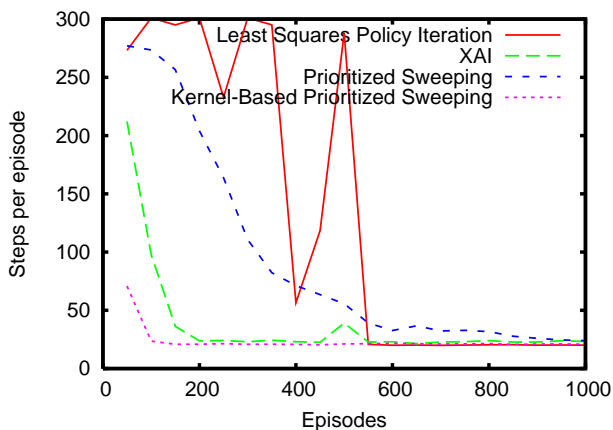


Figure 4: Learning curves for Puddle World

(Lagoudakis & Parr, 2003) is similar to KBRL in that it uses a given sample of transitions to compute the parameters of a function approximator that best approximates the true value function. However, LSPI relies on random exploration and a fixed set of kernels to represent the state space. XAI (eXplore and Allocate, Incrementally) is a method that represents the value function with a network of radial basis functions, allocated online as the agent reaches unexplored regions of the state space (Dutech et al., 2005). It thus resembles KBPS in its instance-based use of Gaussian kernels for approximation, but XAI is a model-free method that uses gradient descent and Sarsa( $\lambda$ ) to update the value function. None of these algorithms achieves the same level of performance as KBPS, which combines instance-based state representation, kernel-based generalization, and model-based exploration.

For the purposes of comparison, we also test an implementation of KBRL without model-based exploration. A simple random exploration mechanism adapts KBRL to the online case. At each time step,

KBRL executes a random action with probability  $0.95^n$ , where  $n$  is the number of completed episodes. After each episode, KBRL recomputes the value function using all the accumulated transition data. However, this algorithm does not perform very well in these benchmark domains: almost every episode times out, until the sample size becomes too large. The agent stumbles upon a goal state very infrequently, and KBRL’s local generalization approach does not allow it to take full advantage of these occasions. Random exploration thus very poorly approximates the assumption of uniform sampling over the state space, which underlies KBRL’s convergence guarantees.

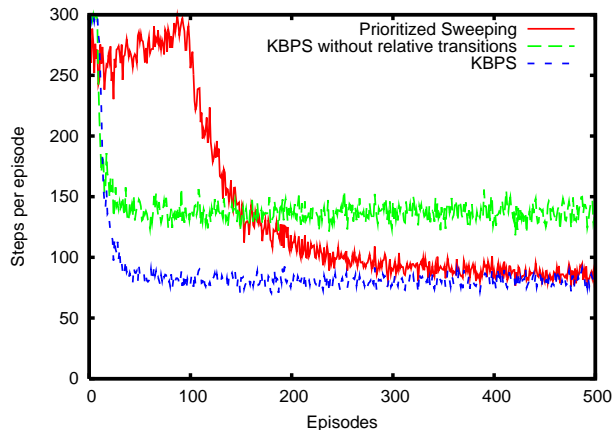
### 4.3. Ablation Study

KBPS benefits from at least two algorithmic contributions: the incorporation of model-based reasoning into KBRL to create an online algorithm (Section 3.1) and the relative transition model (Section 3.2), which removes a source of bias in estimating the model. To measure the magnitude of each benefit, we compare KBPS with standard Prioritized Sweeping and with a version of KBPS lacking the relative transition model. Figure 5 shows the performance of each algorithm, averaged over 50 independent trials in the Mountain Car domain. Our implementation of Prioritized Sweeping uses the same parameters as the finite model-based algorithm submitted to the NIPS workshop: it discretizes each state dimension into 100 intervals and uses  $n_{\text{known}} = 1$ . We use the same parameters described in Section 4.2.1 for KBPS.

The more straightforward combination of KBRL and Prioritized Sweeping converges much more quickly than discrete Prioritized Sweeping, but at the expense of converging to suboptimal policies. Further experimentation has shown that decreasing the bandwidth parameter improves the average quality of the final policy but quickly increases the sample complexity of the algorithm. Adding the relative transition model preserves fast convergence while achieving near-optimal policies in this domain.

## 5. Discussion and Related Work

KBPS learns efficiently in continuous state spaces by combining the advantages of robust function approximation and model-based exploration. This algorithm is not the first to learn models for continuous RL problems, but to our knowledge prior work relies on the assumption of deterministic system dynamics. With this assumption, the successor state becomes a function of the initial state and action, and standard supervised learning techniques may be used to learn the transition



**Figure 5:** Learning curves for Mountain Car. Each curve is the average of 50 independent trials.

function. One such method closely related to KBRL is locally weighted regression (Atkeson et al., 1997), which also uses a kernel to weight previous transitions to estimate the effect of an action at a given state. However, this approach uses the weighted data as inputs to a regression problem, yielding a single average successor. Regression is suitable when nondeterminism only results from noise, but in the general case it will predict only one of possibly many outcomes. In many cases, the average outcome may also be an unlikely or impossible one.

One noteworthy limitation of KBPS is its scalability. As an instance-based algorithm, its time and space complexity grow with the amount of data it collects. Our current implementation simply stops adding data after reaching a fixed threshold, but a more principled approach would strive to keep the most useful samples without introducing significant bias. The relative transition model may help address this problem, since it decouples the finite state space  $D$  from the states that appear in the transition data. Hence, KBPS could stop adding states to  $D$  if the model grows too large to update the value function efficiently, and it can also stop adding transitions to the pseudodata if the dataset grows too large to update the model efficiently.

KBPS is also vulnerable to the curse of dimensionality. As the dimensionality grows, exponentially more data is required to explore each neighborhood of the state space. One solution is to select or to learn more sophisticated kernels that permit generalization of a stored transition beyond its local neighborhood. For example, in real-world domains, many actions are independent of some subset of the state dimensions, and the approximation of those actions can thus generalize freely over those dimensions. The relative transition

model may also play an important role here, since the algorithm must learn that an action leaves an independent dimension unchanged, instead of changing it to a previously observed value.

## 6. Conclusion

We showed how Kernel-Based Reinforcement Learning, an offline function-approximation method proven to converge to the true value function, is equivalent to planning with a finite model approximating the continuous domain. This equivalence enables the synthesis of KBRL and model-based exploration. To achieve a practical algorithm, we proposed a modification to the KBRL transition model that eliminates an important source of bias for many real-world applications. Finally, we demonstrated the efficiency of the resulting algorithm, Kernel-Based Prioritized Sweeping, on standard benchmark domains.

## References

- Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning for control. *Artificial Intelligence Review*, 11, 75–113.
- Dutech, A., Edmunds, T., Kok, J., Lagoudakis, M., Littman, M., Riedmiller, M., Russell, B., Scherrer, B., Sutton, R., Timmer, S., Vlassis, N., White, A., & Whiteson, S. (2005). Reinforcement learning benchmarks and bake-offs II. <http://www.cs.rutgers.edu/~mlittman/topics/nips05-mdp/bakeoffs05.pdf>.
- Kearns, M., & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 260–268).
- Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, 4, 1107–1149.
- Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13, 103–130.
- Ornstein, D., & Sen, S. (2002). Kernel-based reinforcement learning. *Machine Learning*, 49, 161–178.
- Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc.
- Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems* 8.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
- Watkins (1989). *Learning from delayed rewards*. Doctoral dissertation, University of Cambridge, England.