Journal of Artificial Intelligence Research 49 (2014) 705-731 Submitted 12/13; published 4/14

# Convergence of a Q-learning Variant for Continuous States and Actions

**Stephen Carden** CARDEN@CLEMSON.EDU
*Department of Mathematical Sciences,*
*Clemson University*

## Abstract

This paper presents a reinforcement learning algorithm for solving infinite horizon Markov Decision Processes under the expected total discounted reward criterion when both the state and action spaces are continuous. This algorithm is based on Watkins' Q-learning, but uses Nadaraya-Watson kernel smoothing to generalize knowledge to unvisited states. As expected, continuity conditions must be imposed on the mean rewards and transition probabilities. Using results from kernel regression theory, this algorithm is proven capable of producing a Q-value function estimate that is uniformly within an arbitrary tolerance of the true Q-value function with probability one. The algorithm is then applied to an example problem to empirically show convergence as well.

## 1. Introduction

A Markov Decision Process (MDP) is a stochastic control problem, usually considered in discrete time. At each time step, the system is in a state and a controller chooses an action. The system then transitions to a new state with a probability distribution that depends both on the previous state and the action utilized. A reward (or cost, for some authors) is also received. The rewards are allowed to be random variables with a distribution that depends on the state and the action utilized. The goal is to find a policy (a function that maps states to actions) that maximizes rewards (or minimizes costs) according to some measure of goodness. We will consider MDPs for which there are no terminal, absorbing states. In this case, a common optimality criterion is the expected total discounted reward. MDPs under this criterion on finite state-action spaces have been well studied. If the state transition probabilities and reward means are known, then dynamic programming methods can obtain an optimal policy in a finite number of steps (Ross, 1992). However, if probabilities and rewards are initially unknown but transitions and rewards can be simulated or observed, then the problem becomes much more difficult, one amenable to the methods of reinforcement learning.

Watkins (1989) developed Q-learning, a novel and popular algorithm for learning the value of a state-action pair under an optimal policy without explicitly calculating transition probabilities and mean rewards. The values can then be used to construct an optimal policy. If there are a finite number of states and actions, then the learned values are proven to converge to their true values. However, problems with a very large number of states or actions may cause the values to converge unreasonably slowly. Furthermore, if the number of state-action pairs is infinite, then the convergence guarantee no longer holds. A natural question is whether the knowledge gained from an observation can be generalized using function approximators to similar states and actions, and whether the state-action values still converge, either theoretically or empirically. The immediacy of this question is apparent even in Watkins' thesis (1989), where a form of function approximation, the CMAC (Albus, 1975), is used in the solution to two example problems with a large number of state-action pairs.

Function approximators in reinforcement learning may cause value estimates to converge to suboptimal values (Bertsekas, 1995), oscillate (Gordon, 1996), or even diverge (Fairbank & Alonso, 2012; Tsitsiklis & Van Roy, 1996). However, when used carefully, they have also been impressively effective. A famous example of success with a large yet finite state-action space is the TD-Gammon program (Tesauro, 1995), which used temporal difference learning (Sutton, 1988) and a neural network to generalize knowledge. It eventually learned to be competitive with tournament-level human backgammon players. Scenarios with continuous states and finite actions have been well studied. Empirical convergence has been observed with CMACs (Sutton, 1996), neural networks (Rummery & Niranjan, 1994; ten Hagen, 1991), and regression trees (Ernst et al., 2005). Theoretically, there have been positive results as well. A parametric function approximator can yield convergent results if it satisfies certain interpolation properties (Szepesvári & Smart, 2004). A non-parametric technique, kernel regression, has also been proven to converge to optimal values (Ormoneit & Sen, 1999).

Problems where both the states and the actions are allowed to be continuous have been less well-studied. One major reason is that constructing a policy requires taking a supremum over all possible actions, which is straightforward when actions are finite but generally difficult when continuous. This difficulty, though present, has not prevented researchers from developing techniques to make the process manageable. One method is to calculate values for a discrete set of actions, then use a weighted average to produce a continuous valued action (Millán et al., 2002). Baird and Klopf (1993) specifically designed a function approximator, termed "wire-fitting," such that the supremum over the action space can be immediately obtained. Wire-fitting has been combined with neural networks to learn thruster-control in order to drive a submersible to a target with success (Gaskett et al., 1999). Algorithms allowing for stochastic policies can rapidly choose actions from a probability distribution, as in Sequential Monte Carlo Learning (Lazaric et al., 2008).

Despite the practical success of these algorithms, there has been very little theoretical investigation into the convergence properties when actions are continuous. The purpose of this paper is to present what is, to my knowledge, the first off-policy Q-learning variant allowing for continuous state and actions which is proven to converge to optimal values with probability one. Specifically, if the state and action spaces are both allowed to be infinite but compact subsets of Euclidean space, then with a sufficiently small kernel regression bandwidth and suitable continuity conditions on the random rewards and transition probabilities, one may obtain a Q-value function estimate that is uniformly within an arbitrary tolerance of the true Q-value function. Though the intention of the algorithm is to fill a theoretical gap, the results of a proof-of-concept example implementation will be provided.

The particular method of knowledge generalization is Nadaraya-Watson kernel regression, which is a memory-based non-parametric smoothing technique with well-studied properties. It is similar to the function approximators used by Ormoneit and Sen (1999) and Santamaría et al. (1996). A distinction should be made between Nadaraya-Watson kernel regression and the use of kernel functions through the so-called "kernel trick" in the context of reinforcement learning. The methods are similar in that they use a kernel function and deliver non-linear regression results as kernel based weighted sums, but the intermediate methodologies are quite different. See the work of Taylor and Parr (2009) for a discussion of kernelized value function methods, and the work of Xu et al. (2007) for an application of kernelized regression to policy iteration.

In Section 2, Markov Decision Processes will be defined. Section 3 introduces Nadaraya-Watson kernel regression and describes an algorithm that uses kernel smoothing to obtain a function that estimates the Q-value of each state-action pair. The main result of this paper, theoretical convergence of a continuous state and action algorithm, is stated in Section 4 along with sufficient conditions for convergence. Section 5 describes the strategy of the proof, proves a set of lemmas, and finally proves the main result. An example implementation along with a discussion of practical difficulties is in Section 6. Finally, Section 7 concludes with a few ideas for extensions.

## 2. Description of a Continuous Markov Decision Process

It is assumed the reader is familiar with the basic theory of discrete Markov Decision Processes. For an introduction, we recommend the texts by Puterman (1994) and Ross (1992). This paper will consider Markov Decision Processes for which the state and action spaces are not discrete, but are compact subsets of Euclidean space. Let $S$, the state space, be a compact subset of Euclidean space of dimension $d_S$ and $\mathcal{B}(S)$ be the Borel $\sigma$-algebra on $S$. Let $A$, the action space, be a compact subset of Euclidean space of dimension $d_A$. Note that $S \times A$ is a subset of $\mathbb{R}^d$ where $d = d_S + d_A$. Elements of this space will be written $(s, a)$ to make the state and action clear, but for all computational purposes they are best regarded as numerical vectors. Transitions between states are governed by a function $P : S \times \mathcal{B}(S) \times A \to \mathbb{R}_+$ satisfying:

- For each $a \in A$ and $B \in \mathcal{B}(S)$, $P(\cdot, B, a) : S \to \mathbb{R}_+$ is a measurable function.
- For each $s \in S$ and $a \in A$, $P(s, \cdot, a) : \mathcal{B}(S) \to \mathbb{R}_+$ is a probability measure on $S$.
- For each $s \in S$ and $B \in \mathcal{B}(S)$, $P(s, B, \cdot) : A \to \mathbb{R}_+$ is a measurable function.

For each state and action $(s, a)$ there is a random reward $R(s, a)$ bounded by a constant $C_0$.

The Markov Decision Process proceeds as follows: the process begins in some state $s \in S$ and the controller chooses an action $a \in A$. A random reward $R(s, a)$ is received and the system transitions to a new state in accordance with the probability measure $P(s, \cdot, a)$. The system progresses to the next time step, and this scenario repeats from the new state. The reward distributions and transition probabilities are assumed to have the Markov property: they only depend on the current state and action; they do not depend on the history of previous states and actions.

A policy $\pi$ is a measurable function $\pi : S \to A$. In some settings, policies are allowed to be stochastic or non-stationary, but we will restrict our attention to deterministic policies. Define $P_\pi(s, B) := P(s, B, \pi(s))$. Then $P_\pi$ is a transition kernel for a Markov chain on the space $S$.

Let $\gamma, 0 < \gamma < 1$, be a discount factor. Let $(s_t, a_t)$ be the (random) state and action at time $t$. Under a policy $\pi$, the value of a state $s$ is defined under the expected total discounted reward criterion:

$$V^\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) | s_0 = s\right].$$

A policy $\pi^*$ is optimal if it satisfies $V^{\pi^*}(s) = \sup_\pi V^\pi(s), \forall s \in S$.

Our goal is to find an algorithm which will, under suitable conditions, converge to an optimal policy. To that end, we will define *Q-values* as

$$Q^\pi(s,a) := E[R(s,a)] + \gamma \int_S V^\pi(t) P(s, dt, a).$$

This is a continuous version of the discrete definition made by Watkins (1989). Intuitively $Q^\pi(s,a)$ is the value obtained if we start in state $s$, utilize action $a$, and then follow policy $\pi$ thereafter. Consider the Q-values associated with an optimal policy $\pi^*$: $Q^*(s,a) := Q^{\pi^*}(s,a)$. If we can learn the values $Q^*(s,a)$ for all $(s,a)$, then an optimal policy can easily be recovered by setting

$$\pi^*(s) = \underset{a \in A}{\operatorname{argsup}}\, Q^*(s,a).$$

## 3. Description of Algorithm

This section will introduce the form of the function approximator to be used, and then describe an algorithm which will generate a sequence of estimates of the optimal Q-values.

### 3.1 Nadaraya-Watson Kernel Regression

Nadaraya-Watson kernel regression (Watson, 1964; Nadaraya, 1964) is a smoothing technique which estimates the value at a point as a weighted average of nearby observations, using a non-negative kernel function to assign weights to observations based on their distance. Kernels typically are symmetric, peak at zero, and decrease away from zero so that the weight of an observation is inversely related to the distance. Formally, we will require $K : \mathbb{R}^d \to \mathbb{R}_+$ to be a multivariate function satisfying:

KI. K is integrable: $\int_{\mathbb{R}^d} K(u)du < \infty$.

KII. K is bounded: $||K||_\infty < C_K < \infty$.

KIII. K has compact support: There exists $L < \infty$ such that

$$K(u) = 0 \text{ when } ||u||_\infty > L.$$

KIV. K is Lipschitz: There exists $M < \infty$ such that for all $u, u' \in \mathbb{R}^d$,

$$|K(u) - K(u')| \leq M||u - u'||_\infty.$$

The compact support condition is not required in all applications (indeed, the Gaussian function is a common kernel choice) but will be used here to reduce computation load by assigning zero weight to sufficiently distant observations.

A bandwidth $h > 0$ is a parameter used to fine-tune the level of smoothing. Define

$$K_h(u) := \frac{1}{h} K(u/h).$$

If $u$ is non-scalar, the division may be understood to be component-wise. Values of $h$ which are too large or small relative to the sample size tend to oversmooth or overfit the data. Typically when using kernel regression the bandwidth decreases toward zero as the number of observations increases. A discussion of the particulars of bandwidth selection is beyond the scope of this paper, but a rule of thumb obtained from assuming Gaussian density and unit variance is $h = n^{-\frac{1}{4+d}}$, where $n$ is the number of observations and $d$ is the dimension (Härdle, 2004). This rule of thumb should be regarded as a starting value, not a definitive answer. For example, cross-validation is a common method for selecting bandwidths. For more information, the interested reader is referred to the texts by Silverman (1986) or Simonoff (1996). For the purpose of the algorithm to follow, we need only a constant bandwidth which is sufficiently small.

Given a set of $n$ ordered pairs $(x_i, y_i)$ and a bandwidth $h$, the Nadaraya-Watson estimator is

$$\widehat{m_h}(x) = \frac{\sum_{i=1}^{n} K_h(x - x_i) y_i}{\sum_{i=1}^{n} K_h(x - x_i)}.$$

### 3.2 The Algorithm

We will use the following notation. For $n > 0$,

- $h$ is a fixed bandwidth.
- $\gamma$ is a fixed discount factor.
- $(s_n, a_n)$ is the state-action pair at the beginning of iteration $n$.
- $r_n$ is the sample reward observed at iteration $n$.
- $u_n$ is the state that is transitioned to during iteration $n$.
- $y_{h,n} := r_n + \gamma \sup_{a \in A} \widehat{Q}_{h,n-1}(u_n, a)$, where $\widehat{Q}_{h,n-1}(u_n, a)$ is defined below in (2).

Here is a description of the algorithm.

1. Set the initial Q-value estimate function to be zero everywhere. $\widehat{Q}_{h,0}(s, a) = 0$ for all $(s, a) \in S \times A$. Choose some initial state $s_1$.
2. For $n > 0$, pick an action $a_n$ $\epsilon$-greedily (or according to some other exploration method). Define a function $\alpha_{h,n} : S \times A \to [0, 1]$ by

$$\alpha_{h,n}(s, a) = \begin{cases} \dfrac{K_h((s, a) - (s_n, a_n))}{\sum_{j=1}^{n} K_h((s, a) - (s_j, a_j))} & \text{if } \sum_{j=1}^{n} K_h((s, a) - (s_j, a_j)) \neq 0 \\ 0 & \text{if } \sum_{j=1}^{n} K_h((s, a) - (s_j, a_j)) = 0 \end{cases} \tag{1}$$

and use this to update the Q-value estimate function by setting

$$\widehat{Q}_{h,n}(s,a) := (1-\alpha_{h,n}(s,a))\widehat{Q}_{h,n-1}(s,a) + \alpha_{h,n}(s,a)y_{h,n}. \tag{2}$$

Set $s_{n+1} = u_n$.

Note that as a result of Equation (2), one may show by induction that

$$\widehat{Q}_{h,n}(s,a) = \begin{cases} \dfrac{\sum_{j=1}^{n} K_h((s,a)-(s_j,a_j))y_{h,j}}{\sum_{j=1}^{n} K_h((s,a)-(s_j,a_j))} & \text{if } \sum_{j=1}^{n} K_h((s,a)-(s_j,a_j)) \neq 0 \\ \widehat{Q}_{h,0}(s,a) & \text{if } \sum_{j=1}^{n} K_h((s,a)-(s_j,a_j)) = 0. \end{cases} \tag{3}$$

Equation (2) is how the updates are conceptually performed and has a form familiar to classic Q-learning, but Equation (3) is how all calculations are made. The estimates produced by this algorithm are essentially kernel-smoothed averages of the terms $y_{h,n} := r_n + \gamma \sup_{a\in A} \widehat{Q}_{h,n-1}(u_n,a)$.

**Algorithm 1** Pseudocode for theoretical algorithm

Initialize $h$ = bandwidth value, $m$ = maximum iterations,
$\gamma$ = discount factor, $\epsilon$ = exploration parameter
Initialize $\widehat{Q}_{h,0}(s,a) = 0 \ \forall (s,a)$
Set initial state $s_1$
**for** i=1:m **do**
  r = Uniform(0,1) random value
  **if** $r < \epsilon$ **then** $a_i$ = random action
  **else**
    $a_i = \sup_{a\in A} \widehat{Q}_{h,n-1}(s_i,a)$
  **end if**
  $u_i$ = next state, $r_i$ = reward
  $y_{h,i} := r_i + \gamma \sup_{a\in A} \widehat{Q}_{h,i-1}(u_i,a)$
  $\widehat{Q}_{h,i}(s,a) = \frac{\sum_{j=1}^{i} K_h((s,a)-(s_j,a_j))y_{h,j}}{\sum_{j=1}^{i} K_h((s,a)-(s_j,a_j))}$
  $s_{i+1} = u_i$
**end for**

## 4. Statement of Theorem

Assume the following conditions hold:

AI. The state-action pair to be utilized at the beginning of an iteration (regarded as a random variable due to exploration and random transitions) has a density $f : S \times A \to R$ which is positive everywhere and has uniformly continuous and bounded second derivatives.

AII. Rewards are bounded. There exists a $C_0 < \infty$ such that for any $(s,a) \in S \times A$, $R(s,a) < C_0$.

AIII. The expected values of the rewards are Lipschitz continuous across the state-action space. There exists a $C_r$ such that for all $(s_1, a_1), (s_2, a_2) \in S \times A$,

$$|E[R(s_1, a_1)] - E[R(s_2, a_2)]| \leq C_r ||(s_1, a_1) - (s_2, a_2)||.$$

Also, $E[R(s, a)]f(s, a)$ has uniformly continuous and bounded second derivatives.

AIV. Transition probabilities converge weakly and Lipschitz continuously. Let $g : S \to \mathbb{R}$ be continuous and bounded. There exists a $C_t$ such that for any $(s_1, a_1), (s_2, a_2) \in S \times A$,

$$\left| \int g(u) P(s_1, du, a_1) - \int g(u) P(s_2, du, a_2) \right| \leq C_t ||g(u)||_\infty ||(s_1, a_1) - (s_2, a_2)||$$

Also, $\int g(u) P(s, du, a) f(s, a)$ has uniformly continuous and bounded second derivatives.

**Theorem 1.** *Let assumptions AI. - AIV. and kernel conditions KI. - KIV. hold and $\widehat{Q}_{h,n}(s, a)$ be defined by (2). With probability one, for any $\epsilon > 0$, there exists $h = h(\epsilon)$ and $N = N(h, \epsilon)$ such that for $n > N$*

$$\sup_{(s,a) \in S \times A} |\widehat{Q}_{h,n}(s, a) - Q^*(s, a)| < \epsilon. \tag{4}$$

## 5. Lemmas and Proof of Theorem

The proof strategy is inspired by Watkins' original proof over finite spaces. For the interested reader, we suggest the technical report (Watkins & Dayan, 1992) that followed Watkins' thesis. This method is different from the later approaches to proving convergence of Q-learning, such as those of Tsitsiklis (1994) and Jaakkola et al. (1993), which are variations and extensions of stochastic approximation theory. Although Watkins' method does rely on a stochastic approximation result in a minor way, the key intuition is how Q-learning imitates model estimation.

One of the strengths of Q-learning is that it does not require a model of the system to be estimated or maintained. However, the learned Q-values are the optimal values for a model that can be constructed by appropriately assigning weights to the observed rewards and transitions. Specifically, if the learning parameters used to update value estimates are used to weight reward and transition observations to carefully define an artificial process, then the learned Q-value estimates will be optimal for this artificial process. As more rewards and transitions are observed, the artificial process becomes more similar to the real process, so the optimal Q-values for the two processes become more similar, thus the learned values converge to the optimal values for the real process. Watkins called such an artificial process the Action Replay Process.

The proof strategy can be summarized in the following five steps:

1. For a fixed bandwidth value, define an auxiliary Action Replay Process (ARP). This artificial process is purely a proof device.

2. Show that the function $\widehat{Q}_{h,n}(s, a)$ is the optimal Q-value function for the ARP with corresponding bandwidth.

3. Show that for sufficiently small bandwidths, the rewards and transition probabilities from the ARP become arbitrarily close to those of the original process.

4. Show that two MDPs with similar rewards and transition probabilities have similar optimal Q-values.

5. The optimal values of the ARP converge to the optimal values of the original process. Hence by step 2, the Q-values learned by the algorithm converge to the optimal Q-values.

### 5.1 Construction of the ARP (Action Replay Process)

Now the ARP will be defined. It is a finite length, terminating MDP. Pick a bandwidth value $h$, and keep it fixed throughout the discussion that follows. We suggest thinking of the ARP as a card game. Suppose that one is preparing to run the algorithm from Section 3. Also suppose that we have a deck of index cards. For each iteration of the algorithm, we will write the following values on a card: $< s_k, a_k, u_k, r_k, \widehat{Q}_{h,k}(\cdot), \alpha_{h,k}(\cdot) >$. These values are regarded as random variables depending on the random states, actions, and rewards from the original process rather than fixed values from a particular sample trajectory. We also have a card on which the initial $\widehat{Q}_{h,0}(\cdot)$ is written. With this card on bottom, stack all cards in ascending order by iteration as shown in Figure 1.

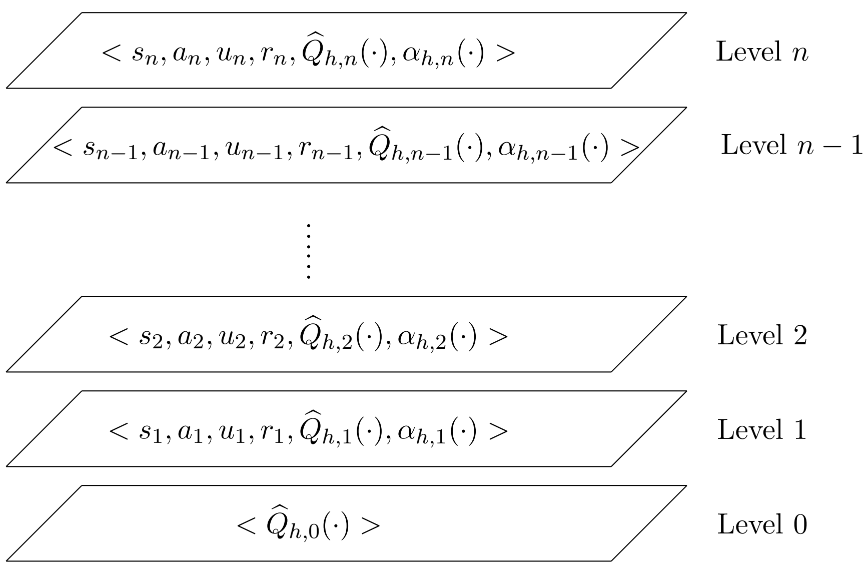


Figure 1: The ARP envisioned as a stack of cards.

**States** A state of the ARP is a 2-tuple $< s, n >$ consisting of a state from the original state space $s \in S$ and a non-negative level $n$ which tells us which card is to be inspected. All cards above level $n$ are ignored, leaving us with a finite stack of cards. Any state with level 0 is a terminal, absorbing state.

**Actions** The actions for the ARP are the same as the actions from the real process, $a \in A$.

**Transitions and Rewards** Suppose the ARP is in state $< s, n >$ and action $a$ is chosen. Inspect the card corresponding to level $n$. With probability $\alpha_{h,n}(s, a)$, we accept this card and receive reward $r_n$ and transition to state $< u_n, n-1 >$. Otherwise, ignore that card and inspect the card at level $n-1$. With probability $\alpha_{h,n-1}(s, a)$, accept this card, receive reward $r_{n-1}$ and transition to state $< u_{n-1}, n-2 >$. Otherwise, continue inspecting cards until one is accepted, where the probability of accepting the card at level $k$ is $\alpha_{h,k}(s, a)$. If the bottom card corresponding to level 0 is reached (that is, the state is $< s, 0 >$ and action $a$ is being utilized) then a reward $Q_{h,0}(s, a)$ is received and the ARP terminates.

As the ARP is an MDP, it has optimal Q-values. Let $Q_h^*(< s, n >, a)$ be the function giving the optimal Q-values for state $< s, n >$ and action $a$ for the ARP with bandwidth $h$.

### 5.2 Lemmas

The optimal Q-values at level $n$ for the ARP corresponding to bandwidth $h$ are the learned Q-value estimates for the original process at iteration $n$ as defined by (2).

**Lemma 1.**

$$Q_h^*(< s, n >, a) = \widehat{Q}_{h,n}(s, a), \forall (s, a) \in S \times A, \forall n \geq 0.$$

*Proof.* Proceed by induction on the level $n$ of the ARP. Recall that if the ARP is in state $< s, 0 >$ and action $a$ is used, then the reward is $\widehat{Q}_{h,0}(s, a)$ and the process terminates. Thus

$$Q_h^*(< s, 0 >, a) = \widehat{Q}_{h,0}(s, a)$$

which proves the $n = 0$ case.

Now suppose by way of induction that $Q_h^*(< s, n-1 >, a) = \widehat{Q}_{h,n-1}(s, a)$ for all states and actions. Let the ARP be in state $< s, n >$ with action $a$ utilized. Recall that by construction of the ARP,

- with probability $\alpha_{h,n}(s, a)$ we receive reward $r_n$ and transition to $< u_n, n-1 >$. Otherwise,
- with probability $1 - \alpha_{h,n}(s, a)$ that card is thrown away and the situation is identical to utilizing $a$ in state $< s, n-1 >$.

Then by conditioning on whether the level $n$ card is accepted or not, we have by the induction hypothesis and (2):

$$\begin{aligned} Q_h^*(< s, n >, a) &= (1 - \alpha_{h,n}(s, a)) Q_h^*(< s, n-1 >, a) \\ &+ \alpha_{h,n}(s, a)(r_n + \gamma \sup_{b \in A} Q_h^*(< u_n, n-1 >, b)) \\ &= (1 - \alpha_{h,n}(s, a)) \widehat{Q}_{h,n-1}(s, a) \\ &+ \alpha_{h,n}(s, a)(r_n + \gamma \sup_{b \in A} \widehat{Q}_{h,n-1}(u_n, b)) \\ &= (1 - \alpha_{h,n}(s, a)) \widehat{Q}_{h,n-1}(s, a) + \alpha_{h,n}(s, a) y_{h,n} \\ &= \widehat{Q}_{h,n}(s, a). \end{aligned}$$

□

The following lemma, which is needed in the proof of Lemma 3, shows that $\sum_{k=1}^{n} \alpha_{h,k}(s,a)$ grows uniformly across the state-action space.

**Lemma 2.** *If assumption AI. holds, then with probability one,*

$$\lim_{n\to\infty} \inf_{(s,a)\in S\times A} \sum_{k=1}^{n} \alpha_{h,k}(s,a) \to \infty. \tag{5}$$

*Proof.* $S \times A$ is compact, so it can be covered by a finite number of disjoint sets of fixed positive diameter with each set having positive Lebesgue measure. Choose the diameter such that $u, u'$ in the same set implies $K_h(u-u') > \delta > 0$. Suppose that $J$ sets are needed, and denote them by $B_1, B_2, ..., B_J$. Set $T_0 = 0$ and for $k > 0$, let $T_k$ denote the first time at which each set has been visited at least $k$ times. Note that by AI. and the Borel-Cantelli Lemma, $P(T_k < \infty) = 1$. Fix $(s,a) \in S \times A$, and letting $C_K$ denote the bound on the kernel values,

$$\sum_{k=1}^{T_n} \alpha_{h,k}(s,a) = \sum_{k=1}^{n} \sum_{j=T_{k-1}+1}^{T_k} \alpha_{h,j}(s,a) \geq \sum_{k=1}^{n} \frac{\delta}{C_K T_k} = \frac{\delta}{C_K} \sum_{k=1}^{n} T_k^{-1}.$$

The terms $T_k, k > 0$, are not independent but can be bounded by random variables which are. Set $W_1 = T_1$. Ignoring all previous visits to sets prior to and at time $T_1$, set $W_2$ to be the number of steps from $T_1$ until each $B_j$ is visited again. It is obvious that $W_2 \geq T_2 - T_1$. Likewise, for $k > 1$, set $W_k$ to be the number of steps from $T_{k-1}$ until each set is visited again, so that $W_k \geq T_k - T_{k-1}$. Notice that $W_1, W_2, ...$ are i.i.d. random variables. Applying these inequalities, one may write

$$\begin{aligned} T_1^{-1} &= W_1^{-1} \\ T_2^{-1} &\geq (W_1 + W_2)^{-1} \\ &\vdots \\ T_k^{-1} &\geq \left( \sum_{j=1}^{k} W_j \right)^{-1}. \end{aligned}$$

We now have

$$\sum_{k=1}^{T_n} \alpha_{h,k}(s,a) \geq \frac{\delta}{C_K} \sum_{k=1}^{n} \left( \sum_{j=1}^{k} W_j \right)^{-1}.$$

It suffices to show that $\sum_{k=1}^{n} \left( \sum_{j=1}^{k} W_j \right)^{-1}$ goes to infinity with probability one. Set $S_n = \sum_{k=1}^{n} W_k$. Then $\sum_{k=1}^{n} \frac{1}{S_k} = \sum_{k=1}^{n} \frac{1}{k} \frac{k}{S_k}$. Since $\frac{k}{S_k} \to \frac{1}{E[W_1]} > 0$, $\sum_{k=1}^{n} \frac{1}{k} \frac{k}{S_k} \to \infty$ with probability one. $(s,a)$ was arbitrary, which proves the result. $\square$

The next lemma shows that if one starts the ARP at a sufficiently high level, the probability of ending below a certain level after a fixed number of actions can be made arbitrarily small.

**Lemma 3.** *Let $\epsilon > 0$, $l_0$ be any level of the ARP, and positive integer $n$ represent the length of a sequence of actions to be followed. Then there exists a level $l > l_0$ such that if the ARP starts at a level above $l$ and follows the sequence of $n$ actions, the probability of ending below level $l_0$ is less than $\epsilon$.*

*Proof.* Proceed by induction. Consider first the case where $n = 1$. For $m > l_0$, suppose the ARP is in state $< s, m >$ and action $a$ utilized. Transitioning to a state with level below $l_0$ means no card at or above level $l_0$ is accepted. The probability of not accepting the card at level $k$ is $1 - \alpha_{h,k}(s, a)$, so the probability of accepting none is $\Pi_{k=l_0}^{m}(1 - \alpha_{h,k}(s, a))$. A well-known inequality from real analysis yields

$$\Pi_{k=l_0}^{m}(1 - \alpha_{h,k}(s, a)) < e^{-\sum_{k=l_0}^{m} \alpha_{h,k}(s,a)}.$$

It follows from Lemma 2 that it is possible to choose $l_1$ such that $m > l_1$ implies

$$\inf_{(s,a)\in S\times A} \sum_{k=l_0}^{m} \alpha_{h,k}(s, a) > -\log \epsilon.$$

Then

$$P(\text{ending below } l_0) = \Pi_{k=l_0}^{m}(1 - \alpha_{h,k}(s, a)) < e^{-\sum_{k=l_0}^{m} \alpha_{h,k}(s,a)} < \epsilon.$$

For a sequence of $n$ actions, there exists $l_1$ such that the probability of ending below $l_0$ from $l_1$ in one transition is less than $\frac{\epsilon}{2}$ and by induction there is an $l_2$ such that the probability of ending below $l_1$ in $n - 1$ transitions is less than $\frac{\epsilon}{2}$. Then if the ARP starts above level $l_2$,

$$\begin{aligned}
&P(\text{below } l_0 \text{ after } n \text{ transitions})\\
=&P(\text{below } l_0 \text{ after } n \text{ transitions}|\text{above } l_1\text{after } n - 1 \text{ transitions})\\
&* P(\text{above } l_1\text{after } n - 1 \text{ transitions})\\
+&P(\text{below } l_0 \text{ after } n \text{ transitions}|\text{below } l_1\text{after } n - 1 \text{ transitions})\\
&* P(\text{below } l_1\text{after } n - 1 \text{ transitions})\\
\leq&\frac{\epsilon}{2} \cdot 1 + 1 \cdot \frac{\epsilon}{2} = \epsilon.
\end{aligned}$$

□

The next lemma allows one to work with finite sequences of actions rather than infinite ones with an arbitrarily small error.

**Lemma 4.** *Consider the value of a state $s$ when a specific sequence of $n$ actions $(a_0, a_1, ..., a_{n-1})$ is to be followed and the process is then terminated:*

$$E\left[\sum_{t=0}^{n-1} \gamma^t R(s_t, a_t)|s_0 = s\right].$$

*Also consider the value of the same state under the same $n$ actions but then followed by an arbitrary policy $\pi$:*

$$E\left[\sum_{t=0}^{n-1} \gamma^t R(s_t, a_t) + \sum_{t=n}^{\infty} \gamma^t R(s_t, \pi(s_t))|s_0 = s\right].$$

*The difference of these two values goes to zero as $n$ increases towards infinity.*

*Proof.* We are interested in the difference

$$E\left[\sum_{t=0}^{n-1}\gamma^t R(s_t,a_t)+\sum_{t=n}^{\infty}\gamma^t R(s_t,\pi(s_t))|s_0=s\right]-E\left[\sum_{t=0}^{n-1}\gamma^t R(s_t,a_t)|s_0=s\right].$$

The difference is clearly the expectation of the second term in the first expectation. Because the rewards are bounded by $C_0$, we can use a change of variables $v=t-n$ and write

$$\begin{aligned}E\left[\sum_{t=n}^{\infty}\gamma^t R(s_t,\pi(s_t))|s_0=s\right]&=E\left[\sum_{v=0}^{\infty}\gamma^{v+n}R(s_v,\pi(s_v))|s_0=s\right]\\&\leq\gamma^n E\left[\sum_{v=0}^{\infty}\gamma^{v}|R(s_v,\pi(s_v))||s_0=s\right]\\&\leq\gamma^n\sum_{v=0}^{\infty}\gamma^v C_0=\gamma^n\frac{C_0}{1-\gamma}\end{aligned}$$

which goes to zero as n goes to infinity. □

The next lemma will consider the reward received from the ARP when in state $<s,n>$ and action $a$ is utilized. This random quantity depends on the state-actions chosen and rewards received in the original process, but it also depends on which levels of the ARP are accepted during state transitions. For the ARP constructed with bandwidth $h$ and rewards received when in state $<s,n>$ and utilizing action $a$, let $\widehat{E}[R_{h,n}(s,a)]$ denote an expectation taken over ARP state transitions, but not over the sequence of state-actions and rewards from the original process. Note that $\widehat{E}[R_{h,n}(s,a)]$ is not a constant, but a random quantity defined on the same sample space as the original process.

Similarly, the probability of the ARP transitioning into a state in set $T\in\mathcal{B}(S)$ is also a random variable depending on which state-actions are chosen in the original process. For the ARP constructed with bandwidth $h$ and starting in state $<s,n>$ and utilizing action $a$, let $\widehat{P}_{h,n}(s,T,a)$ denote the random probability of the ARP transitioning into a state in set $T$.

The next lemma shows that for high enough levels, the rewards and transition probabilities of the ARP are close to the rewards and transition probabilities of the original process.

**Lemma 5.** *If assumptions AI.-AIV. and kernel conditions KI.-KIV. are met, then*

*a) With probability one, for any $\epsilon>0$ there exists an $h=h(\epsilon,T)$ and $N=N(\epsilon,h)$ such that if $n>N$,*

$$\sup_{(s,a)\in S\times A}\left|\widehat{E}[R_{h,n}(s,a)]-E[R(s,a)]\right|<\epsilon.$$

*b) With probability one, for any $\epsilon>0$ and $g:S\to\mathbb{R}$ that is integrable, continuous, and bounded, there exists a $h(\epsilon)$ and $N(\epsilon,h)$ such that if $n>N$,*

$$\sup_{(s,a)\in S\times A}\left|\int_S g(u)\widehat{P}_{h,n}(s,du,a)-\int_S g(u)P(s,du,a)\right|<\epsilon.$$

*Proof.* Consider the expected reward associated with a state-action $(s, a)$ at level $n$ of the ARP. Condition on whether the card at level $n$ is accepted. With probability $1 - \alpha_{h,n}(s, a)$ we do not accept the card, and the expected reward is that of level $n-1$. With probability $\alpha_{h,n}(s, a)$ the level $n$ card is accepted and we receive reward $r_n$. For the special case $n = 0$, we are at the lowest level of the ARP and receive reward $\widehat{Q}_{h,0}(s, a)$. So the expected rewards satisfy the recursive relationship

$$\begin{aligned}
&\widehat{E}[R_{h,0}(s,a)] = \widehat{Q}_{h,0}(s,a), \\
&\widehat{E}[R_{h,n}(s,a)] = (1 - \alpha_{h,n}(s,a))E[R_{h,n-1}(s,a)] + \alpha_{h,n}(s,a)r_n, \text{ for } n > 0.
\end{aligned}$$

Now we show by induction that

$$\widehat{E}[R_{h,n}(s,a)] = \frac{\sum_{j=1}^{n} K_h((s,a) - (s_j,a_j))r_j}{\sum_{j=1}^{n} K_h((s,a) - (s_j,a_j))}$$

when $\sum_{j=1}^{n} K_h((s,a) - (s_j,a_j)) \neq 0$. For the $n = 1$ case,

$$\widehat{E}[R_{h,1}(s,a)] = (1 - \alpha_{h,1}(s,a))\widehat{Q}_{h,0}(s,a) + \alpha_{h,1}r_1.$$

Replace the $\alpha_{h,1}(s, a)$ terms as per the definition in (1).

$$\begin{aligned}
\widehat{E}[R_{h,1}(s,a)] &= \left(1 - \frac{\sum_{j=1}^{1} K_h((s,a) - (s_j,a_j))}{\sum_{j=1}^{1} K_h((s,a) - (s_j,a_j))}\right)\widehat{Q}_{h,0}(s,a) + \frac{\sum_{j=1}^{1} K_h((s,a) - (s_j,a_j))}{\sum_{j=1}^{1} K_h((s,a) - (s_j,a_j))}r_1 \\
&= r_1.
\end{aligned}$$

Now assume the induction hypothesis holds for $n - 1$.

$$\begin{aligned}
\widehat{E}[R_{h,n}(s,a)] &= (1 - \alpha_{h,n}(s,a))\widehat{E}[R_{h,n-1}(s,a)] + \alpha_{h,n}(s,a)r_n \\
&= \left(\frac{\sum_{j=1}^{n-1} K_h((s,a) - (s_j,a_j))}{\sum_{j=1}^{n} K_h((s,a) - (s_j,a_j))}\right)\left(\frac{\sum_{j=1}^{n-1} K_h((s,a) - (s_j,a_j))r_j}{\sum_{j=1}^{n-1} K_h((s,a) - (s_j,a_j))}\right) \\
&+ \frac{K_h((s,a) - (s_j,a_j))}{\sum_{j=1}^{n} K_h((s,a) - (s_j,a_j))}r_n \\
&= \frac{\sum_{j=1}^{n} K_h((s,a) - (s_j,a_j))r_j}{\sum_{j=1}^{n} K_h((s,a) - (s_j,a_j))}.
\end{aligned}$$

The expected rewards for the ARP are equivalent to kernel regression estimates. Assumptions AI., AII., AIII., and kernel conditions KI.-KIV. meet the technical conditions of Hansen (2008). See Theorem 9 on page 735. This yields

$$\sup_{(s,a)\in S\times A} |\widehat{E}[R_{h,n}(s,a)] - E[R(s,a)]| < \epsilon$$

for sufficiently large $n$, which proves part a).

For part b), recall that $u_n$ denotes the state that is transitioned to at iteration $n$. Consider the random variable indicator function for the event that the $n$-th transition from the original process is into $T \in \mathcal{B}(S)$.

$$1_T(u_n) = \begin{cases} 1 & \text{if } u_n \in T \\ 0 & \text{if } u_n \notin T \end{cases}.$$

Similar to rewards, the transition probabilities for the ARP at level $n$ can be conditioned on whether the level $n$ card is accepted or not. Assume we are in state-action $(s, a)$. With probability $1 - \alpha_{h,n}(s, a)$, we do not accept the card and the probability of transitioning into $T$ is that of level $n - 1$. With probability $\alpha_{h,n}(s, a)$ the level $n$ card is accepted and the value of $1_T(u_n)$ tells us whether we transition into $T$ or not. Thus we have the relation

$$\begin{aligned}
\widehat{P}_{h,1}(s, T, a) =& \alpha_{h,1}(s, a)1_T(u_1), \\
\widehat{P}_{h,n}(s, T, a) =& (1 - \alpha_{h,n}(s, a))\widehat{P}_{h,n-1}(s, T, a) + \alpha_{h,n}(s, a)1_T(u_n).
\end{aligned}$$

Let $g : S \to \mathbb{R}$ be a simple function, $g(u) = \sum_{i=1}^{m} a_i 1_{T_i}(u)$. We can apply this relation to the integral of $g$ to find

$$\begin{aligned}
\int g(u)\widehat{P}_{h,1}(s, du, a) &= \sum_{i=1}^{m} a_i \widehat{P}_{h,1}(s, T_i, a) = \sum_{i=1}^{m} a_i 1_{T_i}(u_1) \\
&= g(u_1)
\end{aligned}$$

and

$$\begin{aligned}
\int g(u)\widehat{P}_{h,n}(s, du, a) &= \sum_{i=1}^{m} a_i \widehat{P}_{h,1}(s, T_i, a) \\
&= \sum_{j=1}^{m} a_i \left[(1 - \alpha_{h,n}(s, a))\widehat{P}_{h,n-1}(s, T_i, a) + \alpha_{h,n-1}(s, a)1_{T_i}(u_n)\right] \\
&= (1 - \alpha_{h,n-1}(s, a)) \int g(u)\widehat{P}_{h,n-1}(s, du, a) + \alpha_{h,n}(s, a)g(u_n).
\end{aligned}$$

Using a common argument from measure theory, this relation also holds for any integrable function by approximating with simple functions. The rest of the proof proceeds as in part a), with assumptions AI., AIV., and kernel conditions KI.-KIV. fulfilling the requirements of Hansen's Theorem 9. $\square$

**Definition.** For a given MDP, let $Q(s_1, < a_1, a_2, ..., a_n, t >)$ denote the expected discounted reward received when the process starts in state $s_1$ and actions $a_1, a_2, ..., a_n$ are utilized consecutively and the process then terminates. Formally

$$Q(s_1, < a_1, a_2, ..., a_n, t >) = E\left[\sum_{j=1}^{n} \gamma^{j-1} R(s_j, a_j)\right]$$

where the chance of transitioning to states $s_2, ..., s_n$ is understood to be governed by $P(s_j, \cdot, a_j)$.

The following lemma shows that state-action values possess a sort of continuity. This property will be needed to apply Lemma 7 to state-action values.

**Lemma 6.** *If assumptions AII., AIII., and AIV. hold, then for any $n$, and for any fixed but arbitrary sequence of actions $< a_1, a_2, ..., a_n >$, the function $f : S \to \mathbb{R}$ defined by*

$$f(s) = Q(s, < a_1, a_2, ..., a_n, t >)$$

*is Lipschitz continuous.*

*Proof.* Consider first the case $n = 1$. Let $s_1, s_2 \in S$. Then by AIII.

$$\begin{aligned}&|Q(s_1, < a_1, t >) - Q(s_2, < a_1, t >)|\\ =&|E[R(s_1, a_1)] - E[R(s_2, a_1)]| \leq C_r||(s_1, a_1) - (s_2, a_1)||.\end{aligned}$$

Before considering the case $n > 1$, notice that by assumption AII.

$$\begin{aligned}|Q(s, < a_1, a_2, ..., a_n, t >)| &\leq E\left[\sum_{j=1}^{n} \gamma^{j-1}|R(s_j, a_j)|\right]\\ &\leq \sum_{j=1}^{\infty} \gamma^{j-1} C_0\\ &= \frac{C_0}{1-\gamma}.\end{aligned}$$

Applying AIII. to the difference of means and AIV. to the difference of integrals, we have for $n > 1$ and $s_1, s_2 \in S$

$$\begin{aligned}|f(s_1) - f(s_2)| =&|Q(s_1, < a_1, ..., a_n, t) - Q(s_2, < a_1, ..., a_n, t >)|\\ \leq&|E[R(s_1, a_1)] - E[R(s_2, a_1)]|\\ &+ \left|\gamma \int_S Q(u, < a_2, ...a_n, t >)P(s_1, du, a_1)\right.\\ &\left.- \gamma \int_S Q(u, < a_2, ...a_n, t >)P(s_2, du, a_1)\right|\\ \leq&C_r||((s_1, a_1) - (s_2, a_1))|| + C_t||Q(u, < a_2, ...a_n, t >)||_\infty||(s_1, a_1) - (s_2, a_1)||\\ \leq&C_r||((s_1, a_1) - (s_2, a_1))|| + C_t\frac{C_0}{1-\gamma}||(s_1, a_1) - (s_2, a_1)||.\end{aligned}$$

By taking $C_1 = 2\max\{C_r, \frac{C_t C_0}{1-\gamma}\}$ and inserting into the last inequality, we have

$$|f(s_1) - f(s_2)| \leq C_1||(s_1, a_1) - (s_2, a_1)||.$$

Finally, notice that $||(s_1, a_1) - (s_2, a_1)|| = ||s_1 - s_2||$ from properties of the Euclidean metric. We end with

$$|f(s_1) - f(s_2)| \leq C_1||s_1 - s_2||,$$

which proves the result. □

For the next lemma, suppose one has two MDPs defined on the same state and action space. For $(s,a) \in S \times A$, let $R^{(i)}(s,a)$ and $P^{(i)}(s,\cdot,a)$ denote the rewards and transition probabilities of process $i$ for $i = 1, 2$. The lemma will show that if the expected rewards and transition probabilities of the processes are sufficiently close, then the expected discounted reward of a series of actions is also close.

**Lemma 7.** *Suppose the assumptions required for Lemma 6 hold. For any $\epsilon > 0$ and sequence of actions of length $n$, there exists $\delta_r(n,\epsilon)$ and $\delta_t(n,\epsilon)$ such that if $g : S \to \mathbb{R}$ is continuous and $||g||_\infty \leq \frac{C_0}{1-\gamma}$ and the following two conditions hold:*

*1.*

$$\sup_{(s,a)\in S\times A} \left| E[R^{(1)}(s,a)] - E[R^{(2)}(s,a)] \right| < \delta_r(n,\epsilon)$$

*2.*

$$\sup_{(s,a)\in S\times A} \left| \int_S g(u) P^{(1)}(s,du,a) - \int_S g(u) P^{(2)}(s,du,a) \right| < \delta_t(n,\epsilon),$$

*then for any sequence of actions of length $n$,*

$$|Q^{(1)}(s_1, < a_1, a_2, ..., a_n, t >) - Q^{(2)}(s_1, < a_1, a_2, ..., a_n, t >)| < \epsilon.$$

*Proof.* We proceed by induction on $n$, the number of actions to be executed. For $n = 1$, take $\delta_r(1,\epsilon) = \epsilon$.

$$|Q^{(1)}(s_1, < a_1, t >) - Q^{(2)}(s_1, < a_1, t >)| = |E[R^{(1)}(s_1,a_1)] - E[R^{(2)}(s_1,a_1)]| < \delta_r = \epsilon.$$

Now assume the induction hypothesis holds for a sequence of actions of length $n-1$. Take $\delta_r(n,\epsilon) = \min\left\{\frac{\epsilon}{3}, \delta_r(n-1,\frac{\epsilon}{3})\right\}$, and $\delta_t(n,\epsilon) = \min\left\{\frac{\epsilon}{3}, \delta_t(n-1,\frac{\epsilon}{3})\right\}$. Then for a sequence of length $n$, we can condition on state $s_2$ and write

$$\begin{aligned}
&|Q^{(1)}(s_1, < a_1, ..., a_n, t >) - Q^{(2)}(s_1, < a_1, ..., a_n, t >)| \\
&\leq |E[R^{(1)}(s_1,a_1)] - E[R^{(2)}(s_1,a_1)]| && (6)\\
&+ \gamma \left| \int_S Q^{(1)}(s_2, < a_2, ..., a_n, t >) P^{(1)}(s_1, ds_2, a_1) \right. && (7)\\
&\left. - \int_S Q^{(2)}(s_2, < a_2, ..., a_n, t >) P^{(2)}(s_1, ds_2, a_1) \right|. && (8)
\end{aligned}$$

Add and subtract $\int_S Q^{(1)}(s_2, < a_2, ..., a_n, t >)P^{(2)}(s_1, ds_2, a_1)$:

$$\begin{aligned}
&|Q^{(1)}(s_1, < a_1, ..., a_n, t >) - Q^{(2)}(s_1, < a_1, ..., a_n, t >)| \\
\leq &|E[R^{(1)}(s_1, a_1)] - E[R^{(2)}(s_1, a_1)]| \\
&+ \left| \int_S Q^{(1)}(s_2, < a_2, ..., a_n, t >)P^{(1)}(s_1, ds_2, a_1) \right. \\
&- \left. \int_S Q^{(1)}(s_2, < a_2, ..., a_n, t >)P^{(2)}(s_1, ds_2, a_1) \right| \\
&+ \left| \int_S Q^{(1)}(s_2, < a_2, ..., a_n, t >)P^{(2)}(s_1, ds_2, a_1) \right. \\
&- \left. \int_S Q^{(2)}(s_2, < a_2, ..., a_n, t >)P^{(2)}(s_1, ds_2, a_1) \right|
\end{aligned}$$

From Lemma 6 the value function is continuous, so by assumption the first difference can be made smaller than $\epsilon/3$ by choice of $\delta_r$, the second difference made small by choice of $\delta_t$, and the third difference made small by the induction hypothesis. Each difference is less than $\frac{\epsilon}{3}$, and the result is proved. □

### 5.3 Proof of Theorem

We are now prepared to prove Theorem 1.

*Proof.* The terms with subscripts will denote the values of the ARP with bandwidth $h$. The terms with no subscripts will refer to the original process. For any policy $\pi$, consider the difference

$$\begin{aligned}
&|Q^\pi_h(< s, n >, a) - Q^\pi(s, a)| && (9)\\
\leq &|Q^\pi_h(< s, n >, a) - Q_h(< s, n >, < a, \pi(s_2), ..., \pi(s_m), t >)| && (10)\\
&+ |Q_h(< s, n >, < a, \pi(s_2), ..., \pi(s_m), t >) - Q(s, < a, \pi(s_2), ..., \pi(s_m), t >)| && (11)\\
&+ |Q(s, < a, \pi(s_2), ..., \pi(s_m), t >) - Q^\pi(s, a)|\,. && (12)
\end{aligned}$$

By Lemma 4, $m$ can be chosen large enough such that lines (10) and (12) are less than $\frac{\epsilon}{3}$. By Lemma 5, there exists an $h$ and $N$ such that if more than $N$ iterations are run, the rewards and transition probabilities of the ARP become arbitrarily close to those of the real process. Furthermore, by Lemma 3 there is a higher level $N_1$ such that if one starts at a level above $N_1$, the probability of straying below $N$ after performing $m$ actions is less than $\frac{(1-\gamma)\epsilon}{12C_0}$. When above level $N$, the requirements to apply Lemma 7 are met, and one can make line (11) less than $\frac{2C_0\epsilon}{12C_0-(1-\gamma)\epsilon}$. If the ARP does stray below level $N$, then the difference is bounded by $\frac{2C_0}{1-\gamma}$ as mentioned in Lemma 6. Then by conditioning on whether the ARP strays below level $N$,

$$\begin{aligned}
&|Q_h(< s, n >, < a, \pi(s_2), ..., \pi(s_m), t >) - Q(s, < a, \pi(s_2), ..., \pi(s_m), t >)| \\
\leq &\frac{2C_0}{1-\gamma}\left(\frac{(1-\gamma)\epsilon}{12C_0}\right) + \left(\frac{2C_0\epsilon}{12C_0 - (1-\gamma)\epsilon}\right)\left(\frac{12C_0 - (1-\gamma)\epsilon}{12C_0}\right) = \frac{\epsilon}{3}.
\end{aligned}$$

It follows that

$$|Q_h^{\pi}(< s, n >, a) - Q^{\pi}(s, a)| < \epsilon.$$

Because the policy $\pi$ was arbitrary, we can replace it with a policy optimal for the real process, $\pi^*$. Then we know

$$\left|Q_h^{\pi^*}(< s, n >, a) - Q^*(s, a)\right| < \epsilon.$$

Now we claim that $\pi^*$ also gives optimal values for the ARP. If it did not, and some other policy $\pi_0$ gave higher values, then consider that

$$\left|Q_h^{\pi_0}(< s, n >, a) - Q^{\pi_0}(s, a)\right|$$

can be made arbitrarily small, but this would imply $Q^{\pi_0}(s, a) > Q^{\pi^*}(s, a)$ which contradicts the choice of $\pi^*$ as an optimal policy. We now have

$$|Q_h^*(< s, n >, a) - Q^*(s, a)| < \epsilon.$$

But by Lemma 1, $Q_h^*(< s, n >, a) = \widehat{Q}_h(s, a)$. Thus

$$\left|\widehat{Q}_{h,n}(s, a) - Q^*(s, a)\right| < \epsilon.$$

$(s, a) \in S \times A$ was arbitrary, so the convergence is uniform. $\square$

## 6. Experimental Results

The primary intent of this paper and therefore also this algorithm is to serve as an addition to the theory of continuous domain reinforcement learning by providing asymptotic convergence results. In this section we explore the practical application of the algorithm and show that with minor modification it is capable of obtaining a satisfactory solution to a standard benchmark problem in a reasonable amount of time.

In this section we detail an application to the Mountain Car problem (Moore, 1991). The state space consists of two variables: the position $p$ of the car, constrained to the interval $[-1, 1]$ and the velocity $v$, constrained to $[-3, 3]$. The action space consists of a single variable, a force applied to the vehicle along a line tangent to the road with a value in $[-4, 4]$. The car starts at the bottom of the hill at a standstill, corresponding to an initial state of $(-.5, 0)$. The goal is to drive the car to the top of the hill in the positive direction, i.e., $p > 1$ while keeping the velocity in bounds. If the goal is reached, a reward of 1 is experienced and the trial terminates. If the car goes out of bounds (that is, $p < -1$ or $|v| > 3$) then a reward of $-1$ is received and the trial terminates. In all other states the reward received is zero. The hill is defined by the expression

$$H(p) = \begin{cases} p^2 + p & \text{if } p < 0 \\ \frac{p}{\sqrt{1+5p^2}} & \text{if } p \geq 0 \end{cases}.$$

Figure 2 shows the shape of the hill, the initial position, and the location of the goal. This problem is non-trivial because the maximum force in the positive direction is not large enough to move directly to the goal. Instead, the agent must learn to build momentum by alternating acceleration before trying to move to the goal.

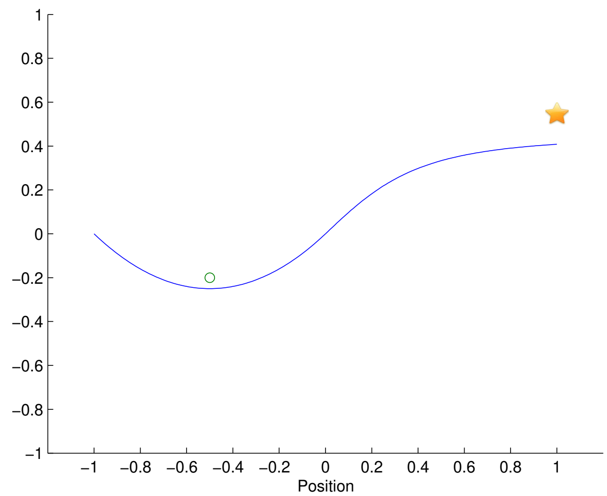


Figure 2: The hill for the mountain car to climb. The circle represents the initial position, and the star represents the goal.

### 6.1 Examination of Assumptions

Assumptions AI.-AIV., while sufficient to ensure convergence, are quite strong. In fact, there are many standard problems in Reinforcement Learning which do not meet them. For example, rewards are often of a discontinuous nature, where most states emit zero reward and crossing a boundary suddenly yields a reward or penalty. The purpose of this section is to show that the proposed algorithm is somewhat robust. That is, even in the presence of violated conditions, it can obtain policies adequate for solving the problem at hand. Specifically, we will see how the Mountain Car problem violates two out of four assumptions, yet the algorithm still produces a policy which can reach the goal state.

The first assumption is the continuous analogue of classic Q-learning's requirement that every state-action pair be visited an infinite number of times. We require the distribution of $(s_n, a_n)$ to possess a density which is positive and sufficiently smooth. However, given the dynamics of the Mountain Car, there are areas of the state space which are inaccessible with the available actions. Consider the state consisting of position $p = .9$ and velocity $v = -2.9$. The position is at the top of the hill near the goal. The maximum acceleration in the negative direction is not strong enough to produce a negative velocity as large as $-2.9$ in the space given.

Figure 3 suggests that the density over the state space is adequately smooth, but it is clear that there are states which have not been visited. However, if the state space is restricted to the set of accessible states, it seems reasonable that the density is everywhere positive.

The second assumption requires bounded rewards. The Mountain Car problem clearly satisfies this.

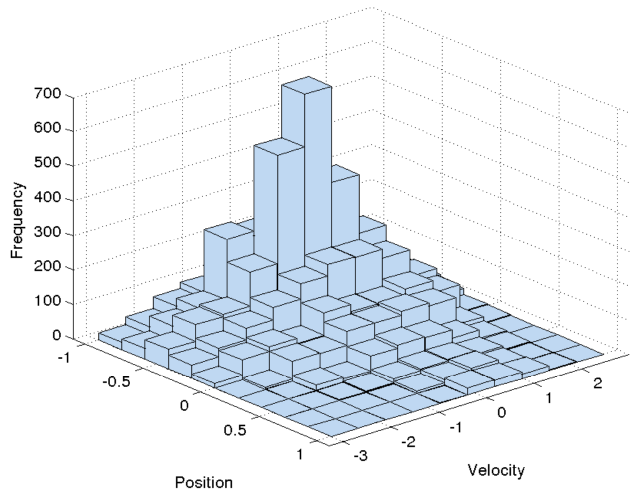


Figure 3: A histogram showing frequency of visits across the state space.

The third assumption requires rewards to be Lipschitz continuous. It is common in Reinforcement Learning for rewards to be zero for most of the state-action space, with a goal state emitting positive rewards and penalties issued for leaving boundaries or entering undesirable states. This is exactly the case for the Mountain Car. One could circumvent this problem by replacing impulse rewards with a low-variance Gaussian density function centered at the goal state, or use a similar method for smoothing rewards without disrupting the overall reward structure.

The fourth assumption requires weak convergence of transition probability measures as states and actions become more similar. Furthermore, that convergence must be uniform across the state-action space. For a problem with smooth, deterministic transitions such as Mountain Car, it is easy to verify convergence in distribution for $u_n$ (the state being transitioned to) as state-actions $(s_n, a_n)$ converge, which is equivalent to weak convergence (see Jacod & Protter, 2003, ch. 18).

It should be noted that the strength of the assumptions are directly linked to the desire for the value function estimate to converge uniformly. Kernel methods can achieve weaker forms of convergence with much weaker assumptions. For example, Devroye and Györfi (1985) show that with proper bandwith selection, kernel density estimates can converge in $L1$ with no conditions on the density being estimated. This suggests that kernel-based algorithms may be able to return good policies despite discontinuities in the reward function, as is the case for the Mountain Car problem. However, this has not been thoroughly investigated for other problems yet.

### 6.2 Computational Considerations

An advantage of non-parametric approximators is that they have the potential to represent any function with arbitrary precision, but at a cost of increased computational load as more observations are used. The following paragraphs will discuss a few suggestions for alleviating this problem.

One idea is to be selective about which observations are kept to use for future calculations while discarding the rest. For example, consider that before a boundary is reached for the first time, all rewards received are equal to zero, which is the initialized function value. Recording these observations will slow down future value function computations but does not add to the knowledge of the system. Suppose that when in state-action $(s, a)$, reward $r_n$ is received and the next state is $u_n$. One may choose to discard this experience if

$$|r_n + \gamma \sup_{b \in A} \widehat{Q}_{h,n-1}(u_n, b) - \widehat{Q}_{h,n-1}(s, a)| < \delta$$

for some tolerance $\delta$. Only keeping experiences with the potential to improve the value function estimate ensures the algorithm learns efficiently.

Another idea is to, at some point, begin discarding old observations as new ones are recorded to keep a fixed number in memory. Consider that for terms $r_n + \gamma \sup_{b \in A} \widehat{Q}_{h,n-1}(u_n, b)$ calculated early, the value $\widehat{Q}_{h,n-1}(u_n, b)$ is not yet calculated with enough data to give a good estimate. Dropping old values in favor of new ones serves a double purpose of capping the increasing computational load and speeding convergence of value estimates. Care should be taken so that no region of the state-action space is left without observations for performing kernel regression.

A third option, and the approach taken in this example, is to "seed" the algorithm with a fixed number of randomly-generated episodes which have reached the goal before beginning the standard $\epsilon$-greedy exploration strategy. Exploration episodes are generated using random action selection. Episodes which reach a penalty state or fail to find the goal within 500 iterations are discarded, while successful episodes are passed to the algorithm. After 20 such episodes, the algorithm chooses random actions with probability $\epsilon$ and optimal (based on current knowledge) actions otherwise. At this point all iterations are passed to the learning algorithm.

As mentioned in the introduction, taking a supremum over all possible actions for a given state is, in general, a difficult problem. For the Mountain Car problem, which has only one action variable, a suitable choice for a kernel function allows for exact solutions. The Epanechnikov kernel is defined by

$$K^e(x) = \begin{cases} \frac{3}{4}(1 - x^2) & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

and is a common choice for kernel regression as it minimizes an approximation of mean integrated square error (see Silverman, 1986, section 3.3.2 for more details). For the three dimensional Mountain Car problem, we define the multivariate kernel as the product of Epanechnikov kernels for each variable. That is, for state-action pairs represented by $(p_k, v_k, a_k)$, we define

$$K((p_1, v_1, a_1) - (p_2, v_2, a_2)) = K^e(p_1 - p_2)K^e(v_1 - v_2)K^e(a_1 - a_2)$$

An advantage of formulating the kernel in this way is that the numerator of the derivative with respect to the action variable is quadratic in the action variable. Because the Epanechnikov kernel is non-zero on a finite interval, the domain of the action variable can be broken into a finite number of intervals on which the quadratic coefficients are constant. Thus all points which are candidate maxima are either a root of one of a finite number of quadratics or breakpoints between intervals. Using this method, the optimal action can be calculated relatively efficiently. If the routine finds multiple actions with the same optimal value, ties are broken randomly. Currently, this strategy works only on problems with one action variable, though the possibility of extending into higher dimensions is being investigated.

### 6.3 Results

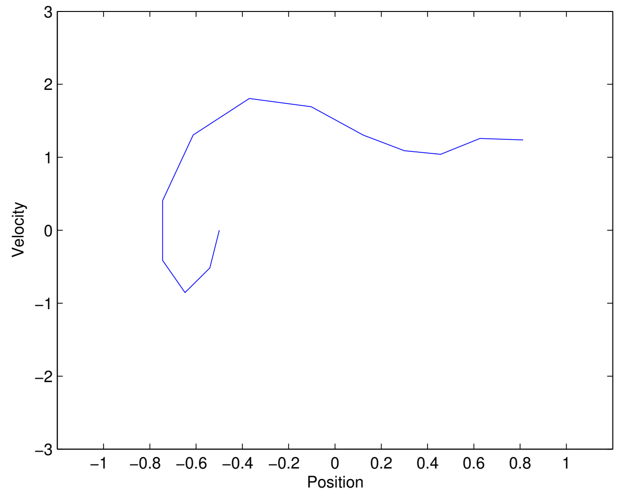


Figure 4: The trajectory through the state space as obtained by the final policy.

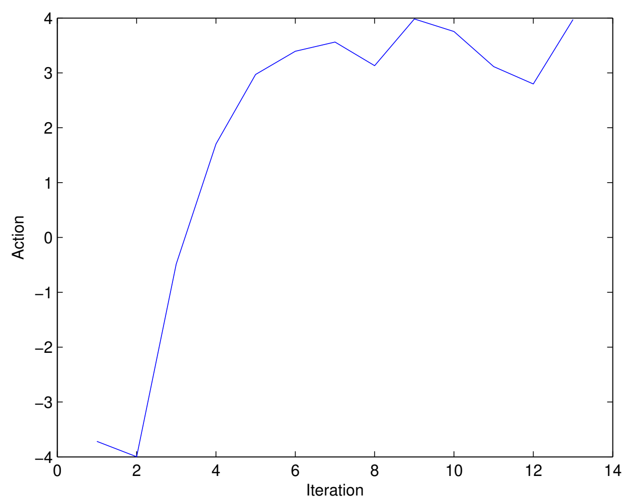


Figure 5: Actions chosen from the final policy.

Implementation was in MATLAB 2012b in Ubuntu 12.04 on hardware with an Intel Xeon 3.47 gigahertz processor and 24 gigabytes of RAM. 7,500 iterations were recorded in 858 seconds. Parameter values were bandwidth $h = .2$, exploration parameter $\epsilon = .9$, discount factor $\gamma = .9$, and $k = 20$ successful episodes to initialize with [1].

The policy obtained is near-optimal. Figure 4 shows the trajectory of the car through the state space. The car begins by accelerating in the negative direction, travels partway up the hill to the left, then changes to positive acceleration until it reaches the goal. Figure 5 shows action selection over time. The optimal policy will choose actions on the extreme ends of the action interval, and the learned policy sometimes chooses actions near but not at the boundary. Figure 6 shows iterations against time. There is an initially surprising phenomena here. One would expect the number of iterations per second to increase as more data is collected, but after around 400 iterations the calculations actually speed up. The reason is that when only a small amount of data is available, the action-selection subroutine will find many actions with the same maximum value. Each candidate best action is kept in memory until the tie is randomly broken at the end of the subroutine. As more data is collected, ties occur less commonly, and the subroutine finishes in less time. For a brief time, the increase in speed from faster action-selection outpaces the decrease in speed from accumulating data. Over time, this phenomena vanishes, and the algorithm slows down once again. Figure 7 shows iterations against time for 25,000 iterations. From this plot it is clear that the described phenomena is temporary.

Another unexpected occurence involved the optimal bandwidth. Repeating the experiment multiple times with different bandwidths revealed that the best results were obtained when the algorithm learned with a bandwidth of .2 but used a smaller bandwidth, .07, when building the final policy. Early trials resulted in a policy that could find the goal but spent more time than necessary building momentum. Upon investigation it was found that the value of state-actions corresponding to the initial state were calculated using values in the initial position but with positive velocity. This caused the policy to use positive acceleration (the best action when velocity is already positive) rather than negative acceleration (the best action when velocity is zero) from the initial state. A smaller bandwidth, specifically .07, avoided this issue. Note that using .07 when the algorithm was in the learning phase did not yield good results because it did not sufficiently generalize when a small amount of data was available. This suggests that even though the convergence proof uses a fixed bandwidth, in practice it is best to decrease the bandwidth as the amount of data increases.

## 7. Conclusion

This paper has presented a reinforcement learning algorithm for continuous states and actions which is proven to converge to an optimal solution. Knowledge generalization is based on Nadaraya-Watson kernel regression. The most similar previous result is that of Ormoneit and Sen (1999), which uses kernel-smoothing for problems with a continuous state space and a finite number of actions. The algorithm presented here is different in a few important ways. First, actions are allowed to be continuous. However, this introduces the burden of ensuring small changes in actions result in small changes to transition dynamics. Second, the assumption on the distribution of states is weaker. Ormoneit and Sen's Assumption

1. For the full source code, see the online appendix associated with this publication.

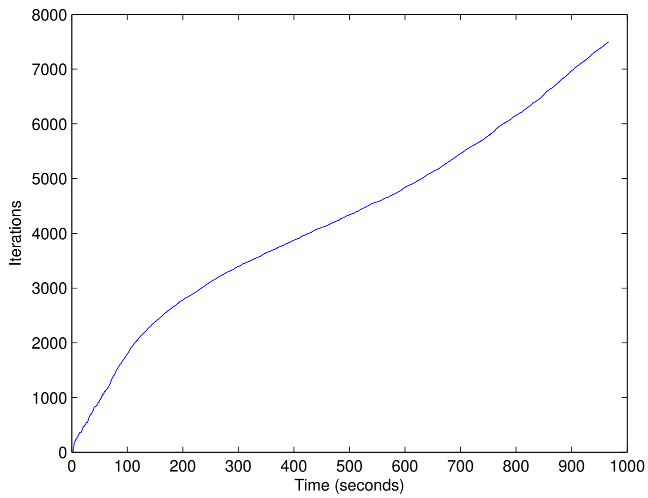


Figure 6: Graph of 7,500 iterations completed over time.

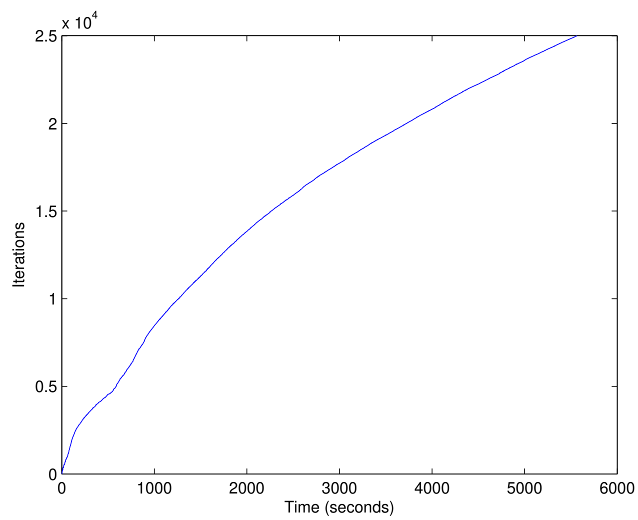


Figure 7: Graph of 25,000 iterations completed over time.

3 requires states to be sampled uniformly from the state space. This has been weakened in the present paper to requiring the distribution of states to have a positive and smooth density.

The presented algorithm is sequential in that it updates after each single observation. The primary reason for considering a sequential rather than batch-mode algorithm is so Watkins' proof strategy can be applied as directly as possible. However, for applications, a batch-mode version possesses many advantages. Informal experiments with the Mountain Car and Inverted Pendulum problems with a batch-mode implementation have produced better results in typically a quarter of the amount of time the sequential algorithm requires.

One of the practical difficulties in implementing the algorithm is that the amount of computation required increases with each recorded observation. In addition to the ideas in Section 6, the author has experimented with sparsifying by selecting a grid of $M$ points $\{(s_j, a_j)|j = 1, ..., M\}$ in the state-action space and assigning to each point a value such that performing kernel regression on those values will yield predictions similar to predictions obtained using kernel regression on the entire set of observations. Essentially, the idea is to use the kernel as a radial basis function at fixed locations in the state-action space and learn the weights for each. Specifically, suppose $N$ transitions have been observed, $N \geq M$, with $\{(s_i, a_i)|i = 1, ...N\}$ and $\{y_{h,i} := r_i + \gamma \sup_{a \in A} \widehat{Q}_{h,i-1}(u_i, a)|i = 1, ..., N\}$ recorded. Define a matrix $X_{N \times M}$ entry-wise by

$$X_{i,j} = \frac{K_h((s_i, a_i) - (s_j, a_j))}{\sum_{j=1}^{M} K_h((s_i, a_i) - (s_j, a_j)}$$

and $\vec{Y}_{N \times 1}$ by $Y_i = y_{h,i}$. Solve for $\vec{\beta}$ that minimizes $||X\vec{\beta} - \vec{Y}||_2$. For a state-action $(s, a)$, define $\vec{K}_{M \times 1}(s, a)$ by $K_j(s, a) = K((s, a) - (s_j, a_j))$. The value of $(s, a)$ can then be estimated by $\vec{K}^T(s, a)\vec{\beta}$. For future observations, update $\vec{\beta}$ as described by Chambers (1971). This method requires computation that is linear in the number of observations and constant in storage requirements.

The fourth assumption concerns weak convergence of state transition probability measures. If the state-action space is separable, it is possible to define a metric on the set of all transition measures such that convergence in the metric is equivalent to weak convergence. See, for example, the Prokhorov metric (Billingsley, 1999). Therefore it is possible to restate assumption four in a more succint manner which is sometimes easier to verify. However, extra work is then required to derive the properties needed in the convergence proof. The author has chosen to state assumption four in the manner that simplifies the proof.

## Acknowledgements

The author would like to thank Peter Kiessler for technical advice, three anonymous referees for many helpful comments, and Tanya Carden and Kris Kelly for proofreading.

## References

Albus, J. S. (1975). A new approach to manipulator control: the cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, *97*, 220–227.

Baird, L. C., & Klopf, A. H. (1993). Reinforcement learning with high-dimensional, continuous actions. Tech. rep. WL–TR-93-1147, Wright-Patterson Air Force Base Ohio: Wright Laboratory.

Bertsekas, D. P. (1995). A counterexample to temporal differences learning. *Neural Computation*, *7*(2), 270–279.

Billingsley, P. (1999). *Convergence of probability measures* (Second edition). Wiley Series in Probability and Statistics: Probability and Statistics. John Wiley & Sons Inc., New York. A Wiley-Interscience Publication.

Devroye, L., & Györfi, L. (1985). *Nonparametric density estimation: the L1 view*. Wiley series in probability and mathematical statistics. Wiley.

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*, 503–556.

Fairbank, M., & Alonso, E. (2012). The divergence of reinforcement learning algorithms with value-iteration and function approximation. In *Proceedings of the IEEE International Joint Conference on Neural Networks*.

Gaskett, C., Wettergreen, D., & Zelinsky, A. (1999). Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, pp. 417–428. Springer-Verlag.

Gordon, G. J. (1996). Chattering in SARSA(lambda) - a CMU learning lab internal report. Tech. rep., Carnegie Mellon University.

Hansen, B. E. (2008). Uniform convergence rates for kernel estimation with dependent data. *Econometric Theory*.

Härdle, W. (2004). *Nonparametric and Semiparametric Models*. Springer Series in Statistics. Springer Berlin Heidelberg.

Jaakkola, T., Jordan, M. I., & Singh, S. P. (1994). Convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, *6*, 1185–1201.

Jacod, J., & Protter, P. (2003). *Probability Essentials*. Universitext (1979). Springer.

Lazaric, A., Restelli, M., & Bonarini, A. (2007). Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. In *Adv. Neural Information Proc. Systems*.

Millán, J. R., Posenato, D., & Dedieu, E. (2002). Continuous-action Q-learning. *Machine Learning*.

Moore, A. (1991). *Efficient Memory-based Learning for Robot Control*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University.

Nadaraya, E. (1964). On estimating regression. *Theory of Probability and its Applications*, *9*, 141–142.

Ormoneit, D., & Sen, S. (1999). Kernel-based reinforcement learning. *Machine Learning*.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience.

Ross, S. (1992). *Applied Probability Models with Optimization Applications*. Dover Books on Mathematics. Dover Publications.

Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. Tech. rep., Cambridge University Engineering Department.

Santamaría, J. C., Sutton, R. S., & Ram, A. (1996). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavor*.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman and Hall, London.

Simonoff, J. (1996). *Smoothing methods in statistics*. Springer, New York.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pp. 1038–1044. MIT Press.

Szepesvári, C., & Smart, W. D. (2004). Interpolation-based Q-learning. In *Proceedings of the International Conference on Machine Learning*, pp. 791–798. ACM Press.

Taylor, G., & Parr, R. (2009). Kernelized Value Function Approximation for Reinforcement Learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 1017–1024.

ten Hagen, S. H. (2001). *Continuous State Space Q-Learning for Control of Nonlinear Systems*. Ph.D. thesis, University of Amsterdam.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Commun. ACM*, *38*(3), 58–68.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. In *Machine Learning*, pp. 185–202.

Tsitsiklis, J. N., & Van Roy, B. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning*, *22*(1–3), 59–94.

Watkins, C. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, University of Cambridge.

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*, 279–292.

Watson, G. S. (1964). Smooth regression analysis. *Sankhya: The Indian Journal of Statistics*, *26*, 359–372.

Xu, X., Hu, D., & Lu, X. (2007). Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, *18*(4), 973–992.