Efficient Value-Function Approximation via Online Linear Regression

Lihong Li and **Michael L. Littman**
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
{lihong,mlittman}@cs.rutgers.edu

Abstract

One of the key problems in reinforcement learning (RL) is balancing exploration and exploitation. Another is learning and acting in large or even continuous Markov decision processes (MDPs), where compact function approximation has to be used. In this paper, we provide a provably efficient, model-free RL algorithm for finite-horizon problems with linear value-function approximation that addresses the exploration-exploitation tradeoff in a principled way. The key element of this algorithm is the use of a hypothesized online linear-regression algorithm in the recently proposed *KWIK* framework. We show that, if the sample complexity of the KWIK online linear-regression algorithm is polynomial, then the sample complexity of exploration of the RL algorithm is also polynomial. Such a connection provides a promising approach to efficient RL with function approximation via studying a simpler setting.

1 Introduction

One of the key problems in reinforcement learning (Sutton & Barto 1998) is the *exploration-exploitation tradeoff*, which strives to balance two competing types of behavior of an autonomous agent in an unknown environment: the agent can either make use of its current knowledge about the environment to maximize its cumulative reward (*i.e.*, to exploit), or sacrifice short-term rewards to gather information about the environment (*i.e.*, to explore) in the hope of increasing future long-term return.

Exploration can be framed as a *dual control* problem, and (in principle) can be solved *optimally* in a Bayesian manner. However, this approach is computationally intractable and it is often not obvious how to select a prior distribution for learning (Duff 2002). We only consider non-Bayesian approaches in this paper. Thrun (1992) surveyed a number of popular exploration rules, including the competence-map approach for continuous MDPs (Thrun & Möller 1992), but little can be said about their performance guarantees. In fact, some of them have provably poor performance in certain situations. Recently, there has been a growing interest in formally analyzing the *sample complexity of exploration* (Kakade 2003) in finite-state Markovian environments. This line of work has significantly advanced understanding of the exploration-exploitation dilemma, but has not been merged with approaches for function approximation needed for scaling up.

In contrast, this paper is concerned with intelligent exploration in large or even continuous environments where compact function approximation has to be used. In particular, we focus on the special case where the value function is represented as a linear combination of predefined features.[1] In contrast to previous work on linear value-function approximation, our algorithm explicitly addresses the question of balancing exploration and exploitation, and we formally analyze its sample complexity. This algorithm works by reducing a finite-horizon reinforcement-learning problem to a sequence of related online linear regression problems, each of which is solved by a hypothesized admissible algorithm, denoted $\mathcal{A}_0$, in the recently proposed KWIK framework (Strehl & Littman 2008). Roughly speaking, $\mathcal{A}_0$ is admissible if it predicts the target value of an example near-accurately except in a polynomial (in relevant quantities that we will make clear) number of examples where it signals "I don't know".

The main contribution of this paper is an efficient *reduction* to KWIK online linear regression from reinforcement learning with linear value-function approximation and an efficient, built-in exploration scheme. This reduction shows how important questions in reinforcement learning, such as the sample complexity of exploration and value-function approximation error, may depend on the related quantities in the simpler setting of KWIK online linear regression. Thus, this connection allows us to study a simpler problem as a means to solve the significantly more difficult RL problem.

Although $\mathcal{A}_0$ is hypothetical right now, a close relative has been successfully created by Strehl & Littman (2008). In particular, the existing algorithm requires that the function to be learned be precisely linear, whereas our algorithm must be tolerant to nearly linear target functions. However, the close relationship between the two problems gives us hope that $\mathcal{A}_0$ can be created under some mild conditions.

[1]Examples of features used in the linear-value-function-approximation context include polynomial features, radial basis functions, and tile coding, etc. (Lagoudakis & Parr 2003; Sutton 1996; Sutton & Barto 1998; van Roy 1998). In this paper, we do not address the problem of selecting or constructing features, which is itself a challenging problem. In practice, good features are often designed by domain experts, and may also be automatically generated (Parr *et al.* 2007).

The rest of the paper is organized as follows. Section 2 defines the KWIK online regression problem and specifies the conditions for the admissible algorithm $\mathcal{A}_0$. Section 3 reviews the RL notation briefly, and then describes the reduction in detail. We provide a few theoretical results including the sample complexity of exploration as well as error bounds for the learned linear value functions, both of which scale up nicely. Finally, Section 4 discusses the relationship between this work to previous results, and Section 5 concludes the paper by pointing out a few research directions.

2 KWIK Online Linear Regression

KWIK stands for "Know What It Knows", and represents a new framework for learning that is particularly well suited for use in RL settings. An essential element of a KWIK learner is that it is able to compute certain quantities to measure how confident it is in its predictions. A simple example is confidence bounds for parameter estimation, which is widely used in statistics and machine learning. Such confidence information is particularly useful for a few purposes. In reinforcement learning, for instance, confidence bounds are used to select actions to guide exploration (Auer 2002; Strehl & Littman 2005).

We first define the *KWIK Online Regression Framework*, adopting terminology from Strehl & Littman (2008). We use $\|\mathbf{x}\|$ to denote the Euclidean norm of a vector $\mathbf{x} \in \mathbb{R}^d$ where $d \in \mathbb{N}$ is the dimension.

Definition 1 *At* timestep $t = 1, 2, 3, \cdots$, *a* KWIK online regression *agent acts according to the following protocol:*

- *First, it receives an* input *vector* $\mathbf{x}_t \in \mathbb{R}^d$.
- *Second, it provides an output* $\hat{y}_t \in [-1, 1] \cup \{\Xi\}$, *where* Ξ *is a special value indicating that the agent is not confident in its prediction and thus refuses to predict a numeric value between* -1 *and* 1. *We call* $\hat{y}_t$ valid *if* $\hat{y}_t \neq \Xi$.
- *Finally, the agent observes the (possibly noisy) ground truth* $y_t \in [-1, 1]$.

The problem becomes a *KWIK online linear regression* problem if we must impose certain assumptions on the functional relation between $\mathbf{x}_t$ and y_t.

Assumption 1 *We make the following assumptions for the KWIK online regression problem:*

A. (Bounded-input assumption) $\|\mathbf{x}_t\| \leq 1$ *for all* t.

B. (Semi-linearity assumption) There exist some (unknown) vector $\mathbf{w}^* \in \mathbb{R}^d$ *and a small number* $\xi \in [0, 1)$ *such that* $\|\mathbf{w}^*\| \leq 1$ *and* $|\mathbf{E}[y_t|\mathbf{x}_t] - \mathbf{w}^* \cdot \mathbf{x}_t| \leq \xi$ *for all* t. *We call the quantity* ξ *the* slack *value.*

Assumption 1A is reasonable as practical problems often have bounded inputs and we can re-scale the inputs so that this assumption holds. Assumption 1B essentially states that the target function being learned is "almost" linear, and the distance to being linearity is measured by the slack value ξ. This assumption is less restrictive than it might appear at the first glance. In practice, with an expanded set of features we can often approximate a learning target by a function linear in the features. This is a common trick to capture nonlinearity via linear functions in many situations including kernel-based learning (Shawe-Taylor & Cristianini 2004).

Note that we make no assumption on the sequence of inputs $\mathbf{x}_t$, except that $\|\mathbf{x}_t\|$ is at most 1. In particular, $\mathbf{x}_t$ can depend on previous inputs $\{\mathbf{x}_1, \cdots, \mathbf{x}_{t-1}\}$, which is fundamentally different from the usual i.i.d. assumption made in supervised-learning problems (*e.g.*, see Hastie, Tibshirani, & Friedman (2003)). As we shall see later, this difference is important when we move to the RL setting in the next section. We next define admissibility of KWIK online regression algorithms.

Definition 2 *A KWIK online regression algorithm* $\mathcal{A}$ *is* admissible *if, for any given* $\epsilon, \delta > 0$, *the following two conditions are satisfied with probability at least* $1 - \delta$:

- *Whenever* $\mathcal{A}$ *predicts a valid* $\hat{y}_t \neq \Xi$, *we have that* $|\hat{y}_t - \mathbf{E}[y_t|\mathbf{x}_t]| \leq \epsilon + \xi$.
- *The number of timesteps* t *for which* $\hat{y}_t = \Xi$ *is bounded by some function* $\zeta(d, 1/\epsilon, 1/\delta)$ *that is polynomial in* d, $1/\epsilon$, *and* $1/\delta$. *We call* ζ *the* sample complexity *of* $\mathcal{A}$.

An admissible, polynomial-time algorithm is proposed recently by Strehl & Littman (2008) for KWIK online linear regression when $\xi = 0$; namely, the target function, $\mathbf{E}[y_t|\mathbf{x}_t]$, is a linear function of $\mathbf{x}_t$. When we allow $\xi > 0$, however, the problem becomes significantly more difficult, as illustrated by the following example.

Example 1 *Fix any* $\xi > \epsilon > 0$. *The input dimension is* $d = 1$, *and the target function* f *to learn is the zero function (which is trivially linear):* $f(x) \equiv 0$. *Let the input at time* t *be* $x_t = \min\{t\beta, 1\}$ *for some small* $\beta > 0$, *then the corresponding output is* $y_t = f(x_t) = 0$. *Assume that the learner knows* f *is noise free. But since it does not know that* f *is exactly linear, it has to predict conservatively to handle the worst-case situations. At time* t *where* $\frac{\xi}{\beta} < t < \frac{1}{\beta}$, *the learner has to say "I don't know" since, based on the training data up to time* $t-1$, *the possible range of* y_t, *which is* $[-\frac{2t\xi}{t-1}, \frac{2t\xi}{t-1}]$, *is too wide to guarantee a prediction error of at most* $\xi + \epsilon$. *By letting* $\beta \downarrow 0$, *the number of* Ξ *does not depend on* ϵ *or* ξ, *and is unbounded.*

It may be appealing to change Definition 2 to tolerate a prediction error of $c\xi + \epsilon$ for some constant $c > 1$. While this modification may allow admissible KWIK online regression algorithms which may be useful on its own, it will lead to an exponential growth of value-function approximation error in the reduction given in Section 3. Therefore, in order to allow for the existence of algorithms that are admissible in the sense of Definition 2, we have to make certain assumptions on the process that generates the data $[\langle \mathbf{x}_t, y_t \rangle]_{t \in \mathbb{N}}$. Such assumptions may place restrictions on, for example, the accumulative loss of the best linear hypothesis, $\mathbf{w}^* \cdot \mathbf{x}$, on the sequence of data. Whether these assumptions are reasonable depends on the applications at hand. For the purpose of this paper, they are denoted abstractly by $\mathcal{C}$.

Assumption 2 *Under certain conditions* $\mathcal{C}$ *on the process generating the samples* $[\langle \mathbf{x}_t, y_t \rangle]_{t \in \mathbb{N}}$, *there exists a KWIK online linear regression algorithm* $\mathcal{A}_0$ *that is admissible and takes polynomial running time in every timestep. We denote its sample complexity by* $\zeta_0(d, 1/\epsilon, 1/\delta)$, *and its per-step computation complexity by* $\tau_0(d, 1/\epsilon, 1/\delta)$.

In the rest of the paper, we assume that condition $\mathcal{C}$ holds whenever we apply the base algorithm $\mathcal{A}_0$.

3 Reinforcement Learning with Linear Value-Function Approximation

Given the background terminology and assumptions in the previous section, we consider reinforcement learning that involves sequential prediction and control. After a brief introduction to notation and terminology, we describe a reduction to KWIK online linear regression from RL.

3.1 Preliminaries

We consider environments modeled as finite-horizon Markov decision processes (Puterman 1994), or MDPs for short. An MDP M can be described by a six-tuple $\langle S, A, T, R, H, \mu_0\rangle$, where S is a set of states, A is a finite set of actions, T is the transition function with $T(s, a, s')$ denoting the probability of reaching s' from s by taking action a, R is a bounded reward function with $R(s, a) \in [0, 1]$ denoting the expected immediate reward gained by taking action a in state s, $H \in \mathbb{N}$ is the horizon, and μ_0 is a start-state distribution.[2] An MDP is said to be *finite* (or *infinite*) if the state space S is finite (or infinite). For convenience, define the set of stages as $[H] = \{1, 2, \cdots, H\}$. An *episode* is a sequence of H state transitions: $\langle s_1, a_1, r_1, s_2, \cdots, s_H, a_H, r_H, s_{H+1}\rangle$, where $s_1 \sim \mu_0$ and s_{H+1} is a terminal state. An agent repeatedly chooses actions until the current episode terminates, and then a new episode starts over again.

A nonstationary policy maps states and stages to actions: $\pi : S \times [H] \to A$. Specifically, $\pi(s, h) \in A$ is the action the agent will take if s is the current state at stage h. Given a policy π, we define the state-value function, $V_h^\pi(s)$, as the expected cumulative reward received by executing π starting from state s at stage h until the episode terminates at stage H. Similarly, the state–action value function (a.k.a. the Q-function), $Q_h^\pi(s, a)$, is the expected cumulative reward received by taking action a in state s at stage h and following π until the episode terminates at stage H. A reinforcement-learning agent attempts to *learn* an optimal policy π^* whose value functions at stage h are denoted by $V_h^*(s)$ and $Q_h^*(s, a)$, respectively. It is known that $V_h^* = \max_\pi V_h^\pi$ and $Q_h^* = \max_\pi Q_h^\pi$. A greedy policy at stage h, denoted π_{Q_h}, with respect to a value function Q_h is one that selects actions with maximum Q-values; namely, $\pi_{Q_h}(s, h) = \arg\max_a Q_h(s, a)$. The greedy policy with respect to Q_h^* is optimal for stage h. The Bellman equation plays a central role to many RL algorithms including the one we will describe: for any $s \in S, a \in A, h \in [H]$,

$$Q_h^*(s, a) = R(s, a) + \sum_{s' \in S} \left(T(s, a, s') \max_{a' \in A} Q_{h+1}^*(s', a') \right)$$

where Q_{H+1}^* is understood to be the zero function.

[2]In general, an H-horizon MDP may have transition probabilities and reward function dependent on the stage. We choose a simpler definition for ease of exposition. The results and analysis in the paper apply to the general case with minor modifications.

Given the complete model of a finite MDP (*i.e.*, the six-tuple), standard algorithms exist for finding the optimal value function and the optimal policy, including linear programming, value iteration, and policy iteration (Puterman 1994). However, if the transition and/or reward functions are unknown, the agent has to learn the optimal value function or policy by interacting with the environment. Algorithms such as Q-learning with ϵ-greedy exploration (Sutton & Barto 1998) do not address the exploration problem efficiently and may be highly inefficient in some domains (Thrun 1992; Koenig & Simmons 1996).

Recently, there has been a growing interest in formally analyzing the efficiency of exploration strategies in *finite* MDPs. For any fixed ϵ, Kakade (2003) defines the *sample complexity of exploration* of an RL algorithm $\mathcal{A}$ to be the number of timesteps t such that the non-stationary policy at time t, $\mathcal{A}_t$, is not ϵ-optimal from the current state s_t at time t (*i.e.*, $V^{\mathcal{A}_t}(s_t) \le V^*(s_t) - \epsilon$). An algorithm $\mathcal{A}$ is then said to be *PAC-MDP* (Probably Approximately Correct in Markov Decision Processes) if, for any $\epsilon > 0$ and $\delta \in (0, 1)$, its sample complexity of exploration is less than some polynomial in $|S|$, $|A|$, $1/\epsilon$, $1/\delta$, and $1/(1 - \gamma)$, with probability at least $1 - \delta$ (Strehl *et al.* 2006). Examples of PAC-MDP algorithms include E^3 (Kearns & Singh 2002), RMAX (Brafman & Tennenholtz 2002), MBIE (Strehl & Littman 2005), and delayed Q-learning (Strehl *et al.* 2006).

3.2 Efficient RL with Linear Value Functions

In this subsection, we propose an algorithm, $\mathcal{A}_H$, for H-horizon reinforcement learning in which a linear value function is used. In particular, we assume a set of d features are predefined: $\phi : S \times A \mapsto [-1, 1]^d$. A Q-function can then be represented compactly by a weight vector $\mathbf{w}_h \in \mathbb{R}^d$ for each $h \in [H]$: $\hat{Q}_h(s, a) = \mathbf{w}_h \cdot \phi(s, a)$. Such a linear approximation scheme is widely used to solve large-scale RL problems (Lagoudakis & Parr 2003; Sutton & Barto 1998).

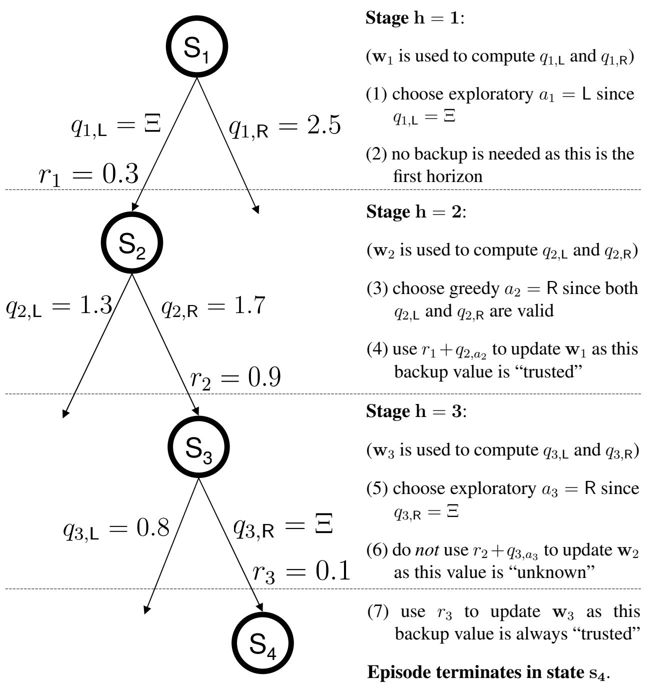
Similarly to the previous section, we make a semi-linearity assumption for the value function at every stage. Remember that outputs in the KWIK online regression (*c.f.*, Definition 1) are in $[-1, 1]$, we need to re-scale the value function to the same range by dividing Q_h^* by H:

Assumption 3 *(Semi-linearity assumption) For every stage $h \in [H]$, there exist some (unknown) vector $\mathbf{w}_h^* \in \mathbb{R}^d$ and a small number $\xi_h > 0$ such that $\|\mathbf{w}_h^*\| \le 1$ and*

$$\left| \frac{Q_h^*(s, a)}{H} - \mathbf{w}_h^* \cdot \phi(s, a) \right| \quad \le \quad \xi_h \qquad (1)$$

for all s and a. Whether it is required to know ξ_h depends on whether such information is needed by $\mathcal{A}_0$.

Algorithm 1 gives a formal description of $\mathcal{A}_H$. Basically, $\mathcal{A}_H$ learns the optimal value functions Q_h^* by treating them as H related KWIK online linear regression problems. It runs H copies of the base algorithm $\mathcal{A}_0$ to update the weight vector $\mathbf{w}_h$ for each stage h. By the Bellman equation, the Q-function at stage h is defined recursively as the sum of immediate reward at stage h and the expected optimal Q-value of the next states. Therefore, the algorithm im-



Stage $\mathbf{h} = 1$:

($\mathbf{w}_1$ is used to compute $q_{1,\mathsf{L}}$ and $q_{1,\mathsf{R}}$)

(1) choose exploratory $a_1 = \mathsf{L}$ since $q_{1,\mathsf{L}} = \Xi$

(2) no backup is needed as this is the first horizon

Stage $\mathbf{h} = 2$:

($\mathbf{w}_2$ is used to compute $q_{2,\mathsf{L}}$ and $q_{2,\mathsf{R}}$)

(3) choose greedy $a_2 = \mathsf{R}$ since both $q_{2,\mathsf{L}}$ and $q_{2,\mathsf{R}}$ are valid

(4) use $r_1 + q_{2,a_2}$ to update $\mathbf{w}_1$ as this backup value is "trusted"

Stage $\mathbf{h} = 3$:

($\mathbf{w}_3$ is used to compute $q_{3,\mathsf{L}}$ and $q_{3,\mathsf{R}}$)

(5) choose exploratory $a_3 = \mathsf{R}$ since $q_{3,\mathsf{R}} = \Xi$

(6) do *not* use $r_2 + q_{3,a_3}$ to update $\mathbf{w}_2$ as this value is "unknown"

(7) use r_3 to update $\mathbf{w}_3$ as this backup value is always "trusted"

Episode terminates in state $\mathbf{s}_4$.

Figure 1: An illustration of the operations of $\mathcal{A}_H$ in a 3-horizon MDP. Two actions are allowed in every state: $\{\mathsf{L}, \mathsf{R}\}$. The same notation as in Algorithm 1 is used.

proves its value-function estimates by performing Bellman-backup-style updates. A central idea behind the efficiency of the reduction is that we only use a backup value when it is "known". A backup value is "known" when the prediction made by $\mathcal{A}_0$ is valid, and thus by Assumption 2 must be near-accurate to the true value. Figure 1 gives a simple H-horizon example for $H = 3$. It illustrates how to choose actions and how to select backup values to do learning.

A quick observation about $\mathcal{A}_H$ is that, if the per-step computation complexity of $\mathcal{A}_0$ is τ_0, then the per-step computation complexity of $\mathcal{A}_H$ is $O(|A|\,\tau_0)$, when execution of lines 17–21 are amortized to every timestep. So, the per-step computation complexity scales nicely from regression problems to sequential decision making in MDPs, in contrast to algorithms such as sparse sampling (Kearns, Mansour, & Ng 2002) that scales exponentially in the horizon length. We next turn to the more difficult questions of sample complexity of exploration and value-function approximation error bounds.

Algorithm 1 Algorithm $\mathcal{A}_H$ for H-horizon reinforcement learning by a reduction to $\mathcal{A}_0$. The base algorithm $\mathcal{A}_0$ is run for each stage h to maintain a separate weight vector $\mathbf{w}_h \in \mathbb{R}^d$ so that the value-function estimate at stage h is $\hat{Q}_h(s,a) = H \cdot \mathbf{w}_h \cdot \phi(s,a)$.

```
 0: Inputs: $A$, $H$, $\phi$, $\epsilon_h$, and $\delta_h$ for $h \in [H]$.
 1: Initialize $H$ copies of $\mathcal{A}_0$, one for each $h \in [H]$. The
    copy at stage $h$ is run with parameters $\epsilon_h$ and $\xi_h$, and is
    denoted by $\mathcal{A}_0^{(h)}$.
 2: for episode $i = 1, 2, 3, \cdots$ do
 3:     for stage $h = 1, 2, 3, \cdots, H$ do
 4:         Observe state $s_h$.
 5:         for all $a \in A$ do
 6:             Use $\mathcal{A}_0^{(h)}$ to compute $q_{h,a} \in [0, H] \cup \{\Xi\}$ as a
                prediction for $Q_h^*(s_h, a)$. Here, if $\mathcal{A}_0^{(h)}$ gives a
                valid prediction, then this prediction has to be
                multiplied by $H$ to obtain $q_{h,a}$ due to the nor-
                malization (1) we have used.
 7:         end for
 8:         if $q_{h,a} = \Xi$ for some $a \in A$ then
 9:             $a_h \leftarrow a$   // do exploration
10:             $L_h \leftarrow$ FALSE   // $Q_h^*(s_h, a_h)$ is "unknown"
11:         else
12:             $a_h \leftarrow \arg\max_a q_{h,a}$   // do exploitation
13:             $L_h \leftarrow$ TRUE   // $Q_h^*(s_h, a_h)$ is "known" and $q_{h,a}$ is "trusted"
14:         end if
15:         Take action $a_h$ and observe reward $r_h$.
16:     end for
17:     for $h = 2, 3, \cdots, H$ do
18:         if $L_h =$ TRUE then
19:             Use $\left(\phi(s_{h-1}, a_{h-1}), \frac{r_{h-1} + q_{h,a_h}}{H}\right)$ as an exam-
                ple for $\mathcal{A}_0^{(h-1)}$ to update $\mathbf{w}_{h-1}$.
20:         end if
21:     end for
22:     Use $\left(\phi(s_H, a_H), \frac{r_H}{H}\right)$ as an example for $\mathcal{A}_0^{(H)}$ to up-
        date $\mathbf{w}_H$.   // terminating rewards are always "trusted"
23: end for
```

Theorem 1 *Suppose Assumption 3 holds. If $\mathcal{A}_0^{(h)}$ is run with parameters ϵ_h and δ_h in Algorithm 1, then:*

I. The number of Ξ outputted in stage $h \in [H]$ is at most

$$\sum_{l=h}^{H} \zeta_0 \left(d, \frac{1}{\epsilon_l}, \frac{1}{\delta_l}\right);$$

II. The total number of Ξ outputted during the whole run of $\mathcal{A}_H$ in all stages is at most

$$\sum_{h=1}^{H} \left(h \cdot \zeta_0 \left(d, \frac{1}{\epsilon_h}, \frac{1}{\delta_h}\right)\right);$$

III. With probability at least $1 - \sum_{l=1}^{H} \delta_l$, all valid Q-value predictions at stage h differ from the true values by at most

$$H \cdot \sum_{l=h}^{H} (\epsilon_l + \xi_l).$$

Before proving this theorem, we first mention a few implications of it. The following corollary, which follows immediately from Theorem 1, indicates that the sample complexity and error bounds of the KWIK online linear regression algorithm $\mathcal{A}_0$ scale nicely to the analogous quantities in the more complicated, H-horizon RL problem.

Corollary 1 *If we let $\epsilon_h = \epsilon_0$, $\delta_h = \delta_0$, and $\xi_h = \xi_0$ for all stage h in Theorem 1, then:*

I. The number of Ξ outputted at stage h is $O\left(H\zeta_0\left(d, \frac{1}{\epsilon_0}, \frac{1}{\delta_0}\right)\right)$;

II. The total number of Ξ outputted during the whole run of $\mathcal{A}_H$ in all stages is $O\left(H^2\zeta_0\left(d, \frac{1}{\epsilon_0}, \frac{1}{\delta_0}\right)\right)$;

III. With probability at least $1 - H\delta_0$, all valid Q-value predictions at stage h differ from the true values by at most $H(H-h+1)(\epsilon_0 + \xi_0) = O(H^2(\epsilon_0 + \xi_0))$.

Using Corollary 1, we can prove the following theorem about the sample complexity of exploration of $\mathcal{A}_H$. Our focus is to provide the first polynomial sample complexity bound, although it is possible to improve the bounds using a more careful analysis.

Theorem 2 *Given any $\epsilon, \delta > 0$, assume we can find a set of d features so that Assumption 3 holds with $\xi_h = O(\frac{\epsilon}{H^3})$. If we run $\mathcal{A}_1^{(h)}$ with $\epsilon_h = O(\frac{\epsilon}{H^3})$ and $\delta_h = \frac{\delta}{2H}$ in Algorithm 1, then the policy used by the agent is ϵ-optimal except*

$$O\left(\frac{H^3}{\epsilon}\zeta_0\left(d, \frac{H^3}{\epsilon}, \frac{H}{\delta}\right)\log\frac{1}{\delta}\right)$$

episodes, with probability at least $1 - \delta$.

3.3 Analysis

Due to space limitation, we only provide proof sketches for the theorems given in the previous subsection. Before proving the sample complexity of exploration bound, we first provide two useful lemmas. The first is simple and the proof is omitted.

Lemma 1 *Let f_1 and f_2 be two real-valued functions on the same finite domain X; namely, $f_i : X \mapsto \mathbb{R}$, for $i = 1, 2$. If $\max_{x\in X}|f_1(x) - f_2(x)| \leq \Delta$ for some $\Delta > 0$, then $|\max_{x\in X} f_1(x) - \max_{x\in X} f_2(x)| \leq \Delta$.*

Lemma 2 *Let π be a policy for an H-horizon MDP. Let s_1 be a fixed start state of an episode, and s_h be the state visited at stage h of this episode. Then,*

$$\begin{aligned} & V_1^*(s_1) - V_1^\pi(s_1) \\ = \;& \mathbf{E}_\pi\left[\sum_{h=1}^{H}\left(Q_h^*(s_h, \pi^*(s_h,h)) - Q_h^*(s_h, \pi(s_h,h))\right)\right], \end{aligned}$$

where $\mathbf{E}_\pi$ stands for the expectation with respect to the probability distributions of trajectories $\rho = [s_1, s_2, \cdots, s_H, s_{H+1}]$ generated by policy π.

PROOF. We let r_h denote the reward received at stage h by following π. Note that both s_h and r_h are random variables whose distributions are completely determined by policy π as s_1 is fixed. Then,

$$\begin{aligned} V_1^*(s_1) &= Q_1^*(s_1, \pi^*(s_1,1)) \\ &= Q_1^*(s_1, \pi(s_1,1)) \\ &\quad + \left(Q_1^*(s_1, \pi^*(s_1,1)) - Q_1^*(s_1, \pi(s_1,1))\right) \\ &= \mathbf{E}_\pi\left[r_1 + V_2^*(s_2)\right] \\ &\quad + \left(Q_1^*(s_1, \pi^*(s_1,1)) - Q_1^*(s_1, \pi(s_1,1))\right). \end{aligned}$$

We apply the derivation above for $V_h^*(s_h)$ recursively up to stage H, and obtain

$$\begin{aligned} V_1^*(s_1) &= \mathbf{E}_\pi[r_1 + r_2 + \cdots + r_H] \\ &\quad + \mathbf{E}_\pi\left[\sum_{h=1}^{H}\left(Q_h^*(s_h, \pi^*(s_h,h)) - Q_h^*(s_h, \pi(s_h,h))\right)\right]. \end{aligned}$$

By definition, $V_1^\pi(s_1) = \mathbf{E}_\pi[r_1 + r_2 + \cdots + r_H]$, which immediately proves the lemma. □

Proof of Theorem 1. The theorem can be proved by mathematical induction. For $h = H$, the theorem is ensured by Assumption 2. For the induction step, assume the theorem holds for all stages $l > h$ where $h < H$ and we consider stage h. Due to operations of Algorithm 1, the transitions from s_h to s_{h+1} in all episodes can be categorized into two groups: (i) $L_{h+1} = \text{TRUE}$, and (ii) $L_{h+1} = \text{FALSE}$.

Transitions belonging to case (i) consist of a stream of data for $\mathcal{A}_0^{(h)}$ to run according to the KWIK online regression protocol defined in Definition 1, and Assumption 2 guarantees that there are at most $\zeta_0(d, 1/\epsilon_h, 1/\delta_h)$ timesteps for which Ξ is outputted. On the other hand, by induction hypothesis, case (ii) happens at most $\sum_{l=h+1}^{H}\zeta_0(d, 1/\epsilon_l, 1/\delta_l)$ times. Therefore, the total number of Ξ outputted in stage h is at most

$$\zeta_0(d, 1/\epsilon_h, 1/\delta_h) + \sum_{l=h+1}^{H}\zeta_0(d, 1/\epsilon_l, 1/\delta_l),$$

which is what we desire to prove for part I.

Part II follows directly from part I.

For part III, the target function to learn at stage h is given by

$$\tilde{Q}_h(s,a) = R(s,a) + \sum_{s'\in S} T(s,a,s')\max_{a'\in A}\hat{Q}_{h+1}(s',a'),$$

where $\hat{Q}_{h+1}$ is the function learned by $\mathcal{A}_H$ in stage $h+1$.[3] By the induction hypothesis, we have $\left|\hat{Q}_{h+1}(s',a') - Q_{h+1}^*(s',a')\right| \leq H\sum_{l=h+1}^{H}(\epsilon_l + \xi_l)$ for all (s',a') whenever Ξ is not outputted. Let $\hat{Q}_h$ be

[3]Strictly speaking, $\hat{Q}_{h+1}$ may change over time and thus $\tilde{Q}_h$ is not a stationary learning target. But, this fact does not affect our analysis as long as $\hat{Q}_{h+1}$ is always bounded between $Q_{h+1}^* - H\sum_{l=h+1}^{H}(\epsilon_l + \xi_l)$ and $Q_{h+1}^* + H\sum_{l=h+1}^{H}(\epsilon_l + \xi_l)$.

the function $\mathcal{A}_0^{(h)}$ learns, then for any (s,a) we have $\left|\hat{Q}_h(s,a) - \tilde{Q}_h(s,a)\right| \le H(\epsilon_h + \xi_h)$ due to Assumptions 2 and 3. Combining all these facts, we have for any (s,a):

$$\begin{aligned}
&\left|\hat{Q}_h(s,a) - Q_h^*(s,a)\right| \\
&\quad\le \left|\hat{Q}_h(s,a) - \tilde{Q}_h(s,a)\right| + \left|\tilde{Q}_h(s,a) - Q_h^*(s,a)\right| \\
&\quad\le H\left(\epsilon_h + \xi_h\right) + \Bigg|\sum_{s'\in S} T(s,a,s')\Big(\max_{a'\in A}\hat{Q}_{h+1}(s',a') \\
&\qquad - \max_{a'\in A} Q_{h+1}^*(s',a')\Big)\Bigg| \\
&\quad\le H(\epsilon_h + \xi_h) \\
&\qquad + \max_{s'\in S}\left|\max_{a'\in A}\hat{Q}_{h+1}(s',a') - \max_{a'\in A} Q_{h+1}^*(s',a')\right| \\
&\quad\le H(\epsilon_h + \xi_h) + H\sum_{l=h+1}^{H}(\epsilon_l + \xi_l) \\
&\quad= H\sum_{l=h}^{H}(\epsilon_l + \xi_l),
\end{aligned}$$

where the last inequality is due to Lemma 1.

Proof for Theorem 2. For episode i, let p_i be the probability of entering some state s for which Ξ is outputted, when start states are drawn from μ_0. Denote by π_i the policy used by $\mathcal{A}_H$ in episode i. Let $\hat{Q}_h$ be the value-function estimate of the algorithm for stage h.

Consider any state trajectory $\rho = [s_1, s_2, \cdots, s_H, s_{H+1}]$ generated by policy π_i. Two situations can occur: (i) Ξ is outputted (maybe multiple times) in ρ, and (ii) Ξ is not outputted in ρ. The probabilities of cases (i) and (ii) are p and $1-p$, respectively. When case (ii) happens, with probability at least $1 - \sum_{h=1}^{H}\delta_h = 1 - \frac{\delta}{2}$, we have for each h,

$$\begin{aligned}
&Q_h^*(s_h, \pi^*(s_h,h)) - Q_h^*(s_h, \pi_i(s_h,h)) \\
&\quad\le Q_h^*(s, \pi^*(s_h,h)) - \hat{Q}_h(s_h, \pi_i(s_h,h)) \\
&\qquad + O\left(H^2(\epsilon_h + \xi_h)\right) \\
&\quad= Q_h^*(s, \pi^*(s_h,h)) - \hat{Q}_h(s_h, \pi_i(s_h,h)) + O\left(\frac{\epsilon}{H}\right) \\
&\quad\le Q_h^*(s, \pi^*(s_h,h)) - \hat{Q}_h(s_h, \pi^*(s_h,h)) + O\left(\frac{\epsilon}{H}\right) \\
&\quad\le O\left(H^2(\epsilon_h + \xi_h)\right) + O\left(\frac{\epsilon}{H}\right) = O\left(\frac{\epsilon}{H}\right), \qquad (2)
\end{aligned}$$

where the first and last inequalities are due to Corollary 1(III), and the second due to the fact that π_i is greedy with respect to $\hat{Q}_h$ when no Ξ is outputted.

For any fixed start state s_1, Lemma 2 asserts that

$$\begin{aligned}
&V_1^*(s_1) - V_1^{\pi_i}(s_1) \\
&\quad= \mathbf{E}_{\pi_i}\left[\sum_{h=1}^{H}\Big(Q_h^*(s_h, \pi^*(s_h,h)) - Q_h^*(s_h, \pi_i(s_h,h))\Big)\right].
\end{aligned}$$

Combined with Equation (2) and the fact that case (i) happens with probability p_i, the equality above implies $V_1^*(s_1) - V_1^{\pi_i}(s_1) = O(\epsilon + Hp_i)$. When $p_i \le p_0$ for some threshold $p_0 = O(\frac{\epsilon}{H})$, we have $V_1^*(s_1) - V_1^{\pi_i}(s_1) = O(\epsilon)$ and also $\mathbf{E}_{s_1\sim\mu_0}\left[V^*(s_1) - V^{\pi_i}(s_1)\right] = O(\epsilon)$, indicating that the policy π_i is indeed $O(\epsilon)$-optimal.

We claim that with high probability $p_i > p_0$ will hold only a polynomial number of episodes. Specifically, Corollary 1 asserts that Ξ is outputted $O(H^2\zeta_0(d, \frac{H^3}{\epsilon}, \frac{H}{\delta}))$ times. Using the inequality of Hoeffding (1963), with probability at least $1 - \frac{\delta}{2}$, the number of episodes i with $p_i > p_0$ is

$$O\left(\frac{H^2\zeta_0(d, \frac{H^3}{\epsilon}, \frac{H}{\delta})}{p_0}\log\frac{1}{\delta}\right).$$

Substituting $p_0 = O(\frac{\epsilon}{H})$ and applying the union bound to the two cases ($p_i > p_0$ and $p_i \le p_0$) gives the lemma.

3.4 An Extension to Discounted RL

While we have focused on finite-horizon RL problems in this paper, it is often easier to model environments by discounted MDPs (Puterman 1994; Sutton & Barto 1998), which are specified by a five-tuple, $\langle S, A, T, R, \gamma\rangle$, where $\gamma \in [0,1)$ is a discount factor. Changes in notation and terminology are necessary since there is no notion of horizon in this setting. Specifically, we only need to consider *stationary* policies and value functions: a policy is a mapping from states to actions: $\pi : S \mapsto A$; the value functions, such as $Q_\gamma^\pi(s,a)$ and $Q_\gamma^*(s,a)$, are defined as the expected cumulative γ-*discounted* reward.

An observation for discounted MDPs is that rewards in the future are exponentially down-weighted. Since rewards are bounded, rewards received after a large number of timesteps contribute little to the value of the current state. Therefore, we may transform a γ-discounted MDP M_γ into an H-horizon MDP M_H so that the optimal value functions of M_H and M_γ differ by at most ϵ, provided

$$H = \Omega\left(\frac{\log\frac{1}{\epsilon(1-\gamma)}}{1-\gamma}\right).$$

It is worth mentioning that even if the optimal value function in M_γ, Q_γ^*, is near linear, the intermediate value functions in M_H, Q_h^*, need not be near linear. We note that this problem may be resolved by using different sets of features at different stages. That is, we require features $\phi_h : S \times A \mapsto [-1,1]^d$ at stage h, and assume that Q_h^* satisfies Assumption 3 with small slack ξ_h. Algorithm $\mathcal{A}_H$ can then be applied.

4 Related Work

Our work in this paper is most related to the original KWIK online linear regression framework proposed by Strehl & Littman (2008). The only difference in problem formulation is that we make a semi-linearity assumption while they assume exact linearity. This change is necessary if we allow Bellman-backup-style updates on the value functions since the backup value (*i.e.*, $\frac{r_{h-1}+q_{h,a_h}}{H}$ in line 19 of Algorithm 1) is unavoidably biased and it is unreasonable to assume the target function at stage $h-1$ remains linear for all possible biases introduced in stage h.

The second significant difference is how KWIK online linear regression is applied to RL problems. Strehl & Littman (2008) adopt a *model-based* approach: they assume the MDP state transitions are governed by a set of linear equations with white, Gaussian noise, and then apply KWIK online linear regression to learn the transition matrices, and finally solve the learned MDP model to obtain a policy that either explores or exploits. Even if an MDP can be accurately modelled as a linear system and approximate planning is concerned, however, solving a continuous MDP remains a challenging task (Chow & Tsitsiklis 1989; Kearns, Mansour, & Ng 2002; Kocsis & Szepesvári 2006). In contrast, the *model-free* approach taken in this paper avoids this problem completely by learning the value function directly. With a learned linear value function, finding the greedy action takes only $O(|A|\, d)$ time per step.

Another related work is metric-E^3 (Kakade, Kearns, & Langford 2003), which also addresses the problem of efficient exploration in continuous MDPs. They also use a model-based approach, and develop sample complexity of exploration in terms of the so-called *cover number* of an MDP—a number that describes how complex the MDP is. Similarly, they also make an assumption on the availability of an efficient, continuous MDP solver, which might limit the use of their algorithm in practice for the same reason.

The KWIK online linear regression framework we described is related to the online learning model of linear functions, where a rich set of beautiful results have been established in the last two decades (*e.g.*, the works by Bernstein (1992), Cesa-Bianchi, Long, & Warmuth (1996), Kivinen & Warmuth (1997), Klasner & Simon (1995), Littlestone, Long, & Warmuth (1995), Long (1997), and Vovk (1998)). In this model, input data are not assumed to be i.i.d. (as what the paper does), and the outputs are roughly a linear function of the inputs. Cumulative absolute and squared error bounds are developed under various assumptions. The main difference between that model and ours is that we require the learner to be aware of the accuracy of its prediction. However, ideas and results in the online learning area may turn out useful for our framework as well.

Recently, Peters & Schaal (2007) proposes an interesting reduction from RL to reward-weighted regression. While they consider the specific task of following a given trajectory in rigid-body systems whose dynamics are governed by a set of equations with unknown parameters, this paper focuses on learning in general MDPs where the learner is not provided with such near-optimal trajectories.

$\mathcal{A}_H$ is similar to delayed Q-learning (Strehl *et al.* 2006) in that both of them allow a value backup to happen only if the backup value is "trusted". In finite MDPs, it is sufficient for delayed Q-learning to become confident by averaging over a large set of samples. In the case of using linear function approximation, $\mathcal{A}_H$ relies on $\mathcal{A}_0$ to do so. In turn, $\mathcal{A}_0$ is expected to resort to more complicated reasoning.

5 Conclusions

The connection between KWIK online regression and reinforcement learning opens a number of interesting directions. First, we are currently developing a concrete algorithm $\mathcal{A}_0$ under mild conditions $\mathcal{C}$, and plan to compare it against the standard RL algorithms with other exploration strategies. With such an algorithm at hand, it is possible to improve the bounds provided in Section 3.2 to make them more practical.

Second, it is important to find more efficient reductions for discounted RL that do not involve an intermediate conversion to an H-horizon problem, as described in Section 3.4. Also, a more careful analysis is needed for the discounted case.

Third, it is observed that a similar reduction in Algorithm 1 applies to KWIK online *nonlinear* regression. This fact allows us to use more expressive classes of nonlinear value functions, which may lead to better policies in some problems.

Finally, we expect fruitful applications of KWIK online regression to (associative) bandit problems (Abe, Biermann, & Long 2003; Auer 2000; 2002; Long 1997). These problems are classic settings for studying the exploration-exploitation dilemma, and have found important applications in the fast growing market of Internet sponsored search (*e.g.*, (Gonen & Pavlov 2007)).

Acknowledgements

We appreciate the helpful discussion with Alex Strehl. The anonymous reviewers also provided constructive comments and links to relevant works that have improved the paper. The work is primarily supported by NSF-ITR-0325281.

References

Abe, N.; Biermann, A. W.; and Long, P. M. 2003. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica* 37(4):263–293.

Auer, P. 2000. An improved on-line algorithm for learning linear evaluation functions. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory (COLT-00)*, 118–125.

Auer, P. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3:397–422.

Bernstein, E. J. 1992. Absolute error bounds for learning linear functions online. In *Proceedings of the Fifth Annual Conference on Computational Learning Theory (COLT-92)*, 160–163.

Brafman, R. I., and Tennenholtz, M. 2002. R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3:213–231.

Cesa-Bianchi, N.; Long, P. M.; and Warmuth, M. 1996. Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Transactions on Neural Networks* 7(3):604–619.

Chow, C.-S., and Tsitsiklis, J. N. 1989. The complexity of dynamic programming. *Journal of Complexity* 5(4):466–488.

Duff, M. O. 2002. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. Ph.D. Dissertation, University of Massachusetts, Amherst, MA.

Gonen, R., and Pavlov, E. 2007. An incentive-compatible multi-armed bandit mechanism. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC-07)*.

Hastie, T.; Tibshirani, R.; and Friedman, J. H. 2003. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 1st edition.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* 58(301):13–30.

Kakade, S.; Kearns, M. J.; and Langford, J. 2003. Exploration in metric state spaces. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, 306–312.

Kakade, S. 2003. *On the Sample Complexity of Reinforcement Learning*. Ph.D. Dissertation, University College London, UK.

Kearns, M. J., and Singh, S. P. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49(2–3):209–232.

Kearns, M. J.; Mansour, Y.; and Ng, A. Y. 2002. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning* 49(2–3):193–208.

Kivinen, J., and Warmuth, M. K. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation* 132(1):1–63.

Klasner, N., and Simon, H. U. 1995. From noise-free to noise-tolerant and from on-line to batch learning. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory (COLT-95)*, 250–257.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *Proceedings of the Seventeenth European Conference on Machine Learning (ECML-06)*, 282–293.

Koenig, S., and Simmons, R. G. 1996. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning* 22(1–3):227–250.

Lagoudakis, M. G., and Parr, R. 2003. Least-squares policy iteration. *Journal of Machine Learning Research* 4:1107–1149.

Littlestone, N.; Long, P. M.; and Warmuth, M. K. 1995. On-line learning of linear functions. *Computational Complexity* 5(2):1–23.

Long, P. M. 1997. On-line evaluation and prediction using linear functions. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory (COLT-97)*, 21–31.

Parr, R.; Painter-Wakefield, C.; Li, L.; and Littman, M. L. 2007. Analyzing feature generation for value-function approximation. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML-07)*, 737–744.

Peters, J., and Schaal, S. 2007. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML-07)*, 745–750.

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

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Strehl, A. L., and Littman, M. L. 2005. A theoretical analysis of model-based interval estimation. In *Proceedings of the Twenty-Second Conference on Machine Learning (ICML-05)*, 857–864.

Strehl, A. L., and Littman, M. L. 2008. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems 20 (NIPS-07)*.

Strehl, A. L.; Li, L.; Wiewiora, E.; Langford, J.; and Littman, M. L. 2006. PAC model-free reinforcement learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML-06)*, 881–888.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Sutton, R. S. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8 (NIPS-95)*, 1038–1044.

Thrun, S., and Möller, K. 1992. Active exploration in dynamic environments. In *Advances in Neural Information Processing Systems 4 (NIPS-91)*, 531–538.

Thrun, S. 1992. The role of exploration in learning control. In White, D. A., and Sofge, D. A., eds., *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold. 527–559.

van Roy, B. 1998. *Learning and Value Function Approximation in Complex Decision Processes*. Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Vovk, V. 1998. Competitive on-line linear regression. In *Advances in Neural Information Processing Systems 10 (NIPS-97)*, 364–370.