

Reducing reinforcement learning to KWIK online regression

Lihong Li · Michael L. Littman

Published online: 29 June 2010
© Springer Science+Business Media B.V. 2010

Abstract One of the key problems in reinforcement learning (RL) is balancing exploration and exploitation. Another is learning and acting in large Markov decision processes (MDPs) where compact function approximation has to be used. This paper introduces REKWIRE, a provably efficient, model-free algorithm for finite-horizon RL problems with value function approximation (VFA) that addresses the exploration-exploitation tradeoff in a principled way. The crucial element of this algorithm is a reduction of RL to online regression in the recently proposed *KWIK* learning model. We show that, if the *KWIK* online regression problem can be solved efficiently, then the sample complexity of exploration of REKWIRE is polynomial. Therefore, the reduction suggests a new and sound direction to tackle general RL problems. The efficiency of our algorithm is verified on a set of proof-of-concept experiments where popular, ad hoc exploration approaches fail.

Keywords Reinforcement learning · Exploration · PAC-MDP · Knows What It Knows (KWIK) · Online regression · Value function approximation

Mathematics Subject Classification (2010) 68T05

Part of this work was done while L. Li was at Rutgers University.

L. Li (✉)
Yahoo! Research, 4401 Great America Parkway,
Santa Clara, CA 95054, USA
e-mail: lihong@yahoo-inc.com

M. L. Littman
Rutgers Laboratory for Real-Life Reinforcement Learning (RL³),
Department of Computer Science, Rutgers University,
Piscataway, NJ 08854, USA
e-mail: mlittman@cs.rutgers.edu

1 Introduction

One of the key problems in reinforcement learning [40] 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, although progresses have been made recently (e.g., [34]), this approach remains computationally intractable in general, and it is often not always obvious how to select an appropriate prior distribution (see, e.g., [9]). We only consider non-Bayesian approaches in this paper, and leave it as future work to combine our approach with Bayesian exploration (see [1] and [21] for two recent advances along this line of work). Thrun [41] surveyed a number of popular exploration rules, including the competence-map approach for continuous MDPs, 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* [15] in finite MDPs. This line of work has significantly advanced our understanding of the exploration-exploitation dilemma, but has not been merged comprehensively with approaches for function approximation needed for scaling up except for model-based approaches that often require a computationally expensive step of solving an MDP [32].

In contrast, this paper is concerned with *model-free* efficient exploration in large or even continuous MDPs where compact function approximation has to be used. Here, value functions are represented compactly from a parameterized function class, a popular example of which is linear VFA that combines predefined features linearly, which is widely used to solve large-scale problems [22, 40]. In contrast to previous work on VFA, 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 RL problem to a series of related online regression problems, each of which is solved by a base algorithm (or “oracle”), denoted \mathcal{A}_0 , in the recently proposed KWIK online regression framework [31, 39]. Roughly speaking, \mathcal{A}_0 is able to predict the target value of an example near-accurately except in a polynomial (in relevant quantities that we will make clear) number of times where it *explicitly* signals “I don’t know”.

The main contribution of this paper is a general *reduction* to KWIK online regression from RL with VFA and an efficient, built-in exploration scheme. Although KWIK online regression is a challenging problem in general,¹ our reduction shows how important questions in RL, 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 regression. Thus, this connection allows us to study a *single-step* prediction problem as a means to approach the more general, *multi-step* decision-making problems in RL. Furthermore, our reduction is provably efficient

¹See [29] for an impossibility result, although an efficient algorithm exists under a slightly different assumption of [39].

and scales polynomially with the horizon of the problem, in contrast to previous reductions that either scales exponentially in the horizon or requires strong prior knowledge (see Section 5 for details).

The rest of the paper is organized as follows. Section 2 defines the KWIK online regression problem. Section 3 reviews the RL notation briefly, and then describes the reduction in detail. Specifically, we provide a few theoretical results including the sample complexity of exploration as well as error bounds for the learned value functions, both of which scale up polynomially in relevant quantities. We then test the algorithm in Section 4 on a few problems where exploration is known to be hard. Finally, Section 5 discusses the relationship between this work to previous results, and Section 6 concludes the paper with a few research directions.

2 KWIK online regression

KWIK stands for “Knows What It Knows”, and represents a new framework for machine learning that is particularly well suited for use in RL settings [31, 39]. 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 intervals 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 intervals have proved useful for efficient exploration [2, 33, 38].

We first define the *KWIK Online Regression* framework; see [31] for the definition of a general KWIK setting. For a d -dimensional vector $\mathbf{x} \in \mathbb{R}^d$, we use $\|\mathbf{x}\|$ to denote the ℓ_2 -norm.

Definition 1 Let \mathcal{F} be a class of real-valued functions such that $f : \mathbb{R}^d \rightarrow [-1, 1]$ for each $f \in \mathcal{F}$. At timestep $t = 1, 2, 3, \dots$, a *KWIK online regression learner* acts according to the following protocol:

1. It receives an *input* vector $\mathbf{x}_t \in \mathbb{R}^d$.
2. It provides an output $\hat{y}_t \in [-1, 1] \cup \{\perp\}$, where \perp 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 $\hat{y}_t \neq \perp$.
3. If $\hat{y}_t = \perp$, the agent observes the (possibly noisy) ground truth $y_t \in [-1, 1]$ with which it can improve its future predictions.

The problem becomes a *KWIK online regression* problem for which we make certain assumptions on the functional relation between \mathbf{x}_t and y_t .

Assumption 1 We make the following assumptions for the KWIK learning process in Definition 1:

- A. (*Bounded-input assumption*) $\|\mathbf{x}_t\| \leq 1$ for all t .
- B. (*Near-realizability assumption*) There exist some (unknown) function $f^* \in \mathcal{F}$ and scalar $\xi \in [0, 1)$ such that $|\mathbf{E}[y_t | \mathbf{x}_t] - f^*(\mathbf{x}_t)| \leq \xi$ for all t . Here, the expectation $\mathbf{E}[\cdot]$ is w.r.t. a probability distribution that depends on the input vector \mathbf{x}_t only.

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 function class \mathcal{F} is rich enough to accurately represent the target function, $f^{\text{target}}(\mathbf{x}) = \mathbf{E}[y|\mathbf{x}]$, and the distance to \mathcal{F} is measured by the slack ξ . In the popular choice of *linear* function class, the assumptions means f^{target} is “almost” linear in the input vectors. This assumption is less restrictive than it might appear at the first glance. In practice, for example, with an expanded set of features we can often reasonably approximate a learning target by a function linear in the features. In fact, this is a common trick to capture nonlinearity via linear approximation in many situations such as kernel-based learning. In the general case where \mathcal{F} may contains *nonlinear* functions, it is expected that the slack value ξ is smaller.

Note that we have not made assumptions 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 $\{\mathbf{x}_1, \dots, \mathbf{x}_{t-1}\}$ and predictions $\{\hat{y}_1, \dots, \hat{y}_{t-1}\}$, which is fundamentally different from the i.i.d. assumption usually made in supervised learning. This difference is important when we study RL in the next section, since visited states generally depend on previous actions taken by an RL learner.

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

Here, the distribution of y_t given \mathbf{x}_t is again fixed beforehand, and the probability statement is over randomization in y_t and in \mathcal{A} (if it is not deterministic).

An admissible, polynomial-time algorithm is proposed recently by [39] for KWIK online linear regression when $\xi = 0$ (the realizable case) and functions in \mathcal{F} are all linear; 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 considerably more difficult [27]. In the present paper, we focus on fundamental relations between reinforcement learning and KWIK online regression. Consequently, we assume access to an oracle algorithm that solves the KWIK online regression problem, and use this algorithm to solve reinforcement learning.

Assumption 2 Let \mathcal{A}_0 be an admissible algorithm for solving the KWIK online regression problem according to Definition 2 that takes polynomial running time in every timestep. We denote its sample complexity by $\zeta_0(d, \frac{1}{\epsilon}, \frac{1}{\delta})$, and its per-step computation complexity by $\tau_0(d, \frac{1}{\epsilon}, \frac{1}{\delta})$.

Finally, we note that the KWIK online regression framework is related to the online learning model [6]. In this model, like ours, input data are not assumed to be i.i.d.. Cumulative absolute and squared prediction 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 its prediction accuracy, which is fundamental to our sample complexity of exploration analysis.

3 Model-free reinforcement learning with value-function approximation

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

3.1 Preliminaries

Environments are modeled by finite-horizon MDPs [35], which 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 $[H] = \{1, 2, \dots, H\}$. An *episode* is a sequence of H state transitions: $\langle s_1, a_1, r_1, s_2, \dots, s_H, a_H, r_H, s_{H+1} \rangle$, 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 policy maps states and stages to actions: $\pi : S \times [H] \rightarrow A$. 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 $s \in S, a \in A, h \in [H]$,

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

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

²In 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 a complete model of a finite MDP (*i.e.*, the six-tuple), standard algorithms exist for finding the optimal value function and the optimal policies [35]. If the transition or reward function is 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 and Boltzmann (a.k.a. “softmax”) rules [40] may be highly inefficient in some domains [20, 41].

Recently, there has been a growing interest in formally analyzing the efficiency of exploration in *finite* MDPs. Specifically, at any time t , an RL algorithm \mathcal{A} is viewed as a non-stationary policy that determines which action a_t to take based on the sequence of interactions (namely, the state–action–reward–next-state tuples) between the agent and the environment. Value functions can be determined in a similar way for non-stationary policies. We denote by $V_h^{A_t}$ the value function of this non-stationary policy in the t -th episode. For any fixed ϵ , [15] defines the *sample complexity of exploration* of an RL algorithm \mathcal{A} to be the number of episodes t in which the non-stationary policy is not ϵ -optimal from the start state; namely, $V_1^{A_t}(s_1) \leq V_1^*(s_1) - \epsilon$. An algorithm \mathcal{A} is then said to be *PAC-MDP* [37] 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 H , with probability at least $1 - \delta$. Examples of PAC-MDP algorithms for finite-state, finite-action MDPs include E^3 [18], R_{MAX} [5], MBIE [38], and delayed Q-learning [37].

3.2 PAC-MDP reinforcement learning with value-function approximation

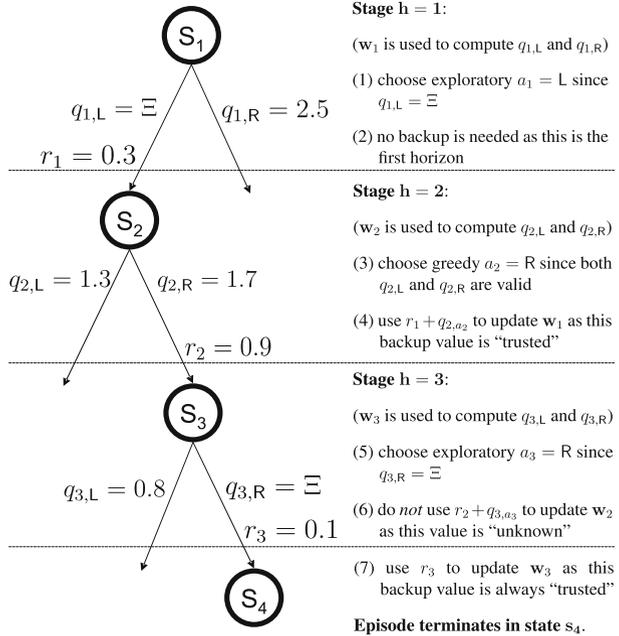
In this subsection, we propose an algorithm, REKWIRE (REinforcement learning based on KWIK online REgression), for H -horizon reinforcement learning in which value function approximations are used. In particular, we assume a set of d features are predefined: $\phi : S \times A \mapsto [-1, 1]^d$. A Q-function can then be represented compactly as a function $f \in \mathcal{F}$ over this feature vector: for each $h \in [H]: \hat{Q}_h(s, a) = f(\phi(s, a))$. In particular, the set of linear value functions is defined by

$$\mathcal{F} = \{ f \mid f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \mathbf{w} \in \mathbb{R}^d \}.$$

Algorithm 1 gives a formal description of REKWIRE. Basically, the algorithm learns the optimal value functions Q_h^* by treating them as H related KWIK online regression problems. It runs H copies of the base algorithm \mathcal{A}_0 to KWIK-learn the optimal value function Q_h^* for all stages h . By the Bellman equation (1), 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 (when a greedy policy is being followed). Therefore, the algorithm improves 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. Figure 1 gives a simple H -horizon example for $H = 3$ and illustrates how REKWIRE chooses actions and computes backup values.

Similarly to the previous section, we need to quantify how expressive the candidate function class \mathcal{F} is to capture the value function to be learned at every stage. First, remember that outputs in the KWIK online regression (c.f., Definition 1) are in $[-1, 1]$, we need to re-scale the value function by $1/H$. Then, let \hat{Q}_{h+1} be the estimate

Fig. 1 An illustration of the operations of REKWIRE in a 3-horizon MDP. Two actions are allowed in every state: {L, R}. The same notation as in Algorithm 1 is used



of Q_{h+1}^* by REKWIRE for stage $h + 1$, and so the target function to be learned at stage h is

$$\tilde{Q}_h(s, a) = R(s, a) + \sum_{s' \in S} T(s, a, s') \max_{a' \in A} \hat{Q}_{h+1}(s', a'). \tag{2}$$

Similar to the previous section, we define the slack variable at stage h by

$$\xi_h = \inf_{f \in \mathcal{F}} \sup_{(s,a) \in S \times A} \left| f(\phi(s, a)) - \frac{\tilde{Q}_h(s, a)}{H} \right|. \tag{3}$$

We note that the quantities ξ_h are used only for analysis, but not by our reduction algorithm.

A quick observation about REKWIRE is that, if the per-step computation complexity of \mathcal{A}_0 is τ_0 , then the per-step computation complexity of REKWIRE 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 state-space discretization [8] that scales exponentially in the dimension of S and sparse sampling [17] that scales exponentially in the horizon H . We next turn to the more difficult questions of bounding the sample complexity of exploration and value-function approximation error.

Theorem 1 Let $\mathcal{A}_0^{(h)}$ be run with parameters ϵ_h and δ_h in REKWIRE, then:

- I. The number of \perp 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 \perp outputted during the whole run of REKWIRE 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 $H \cdot \sum_{l=h}^H (\epsilon_l + \xi_l)$.

Proof We prove Theorem 1 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

Algorithm 1 REKWIRE: An Algorithm 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 KWIK-learn the value function at stage h .

```

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  $i = 1, 2, 3, \dots$  do
3:   for stage  $h = 1, 2, 3, \dots, H$  do
4:     Observe state  $s_h$ .
5:     for all  $a \in A$  do
6:       Use  $\mathcal{A}_0^{(h)}$  to compute  $q_{h,a} \in [0, H] \cup \{\perp\}$  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 normalization (3) we have used.
7:     end for
8:     if  $q_{h,a} = \perp$  for some  $a \in A$  then
9:        $a_h \leftarrow a$  {do exploration}
10:       $L_h \leftarrow \text{FALSE}$  { $Q_h^*(s_h, a_h)$  is “unknown”}
11:    else
12:       $a_h \leftarrow \arg \max_a q_{h,a}$  {do exploitation}
13:       $L_h \leftarrow \text{TRUE}$  { $Q_h^*(s_h, a_h)$  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  $h = 2, 3, \dots, H$  do
18:    if  $L_h = \text{TRUE}$  then
19:      Use  $\left( \phi(s_{h-1}, a_{h-1}), \frac{r_{h-1} + q_{h,a_h}}{H} \right)$  as an example for  $\mathcal{A}_0^{(h-1)}$  to learn.
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 learn. {terminating rewards
   are always “trusted”}
23: end for

```

\perp 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 \perp outputted in stage h is at most

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

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 (2).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 \perp is not outputted. Let \hat{Q}_h be the function $\mathcal{A}_0^{(h)}$ learns, then for any (s, a) we have

$$\left| \hat{Q}_h(s, a) - \tilde{Q}_h(s, a) \right| \leq H(\epsilon_h + \xi_h)$$

due to Assumption 2 and (3). Combining all these facts, we have for any (s, a) :

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

where the last inequality uses Lemma 1 with $f_1 = \hat{Q}_{h+1}$ and $f_2 = Q_{h+1}^*$. □

The following corollary, which follows immediately from Theorem 1, indicates that the sample complexity and error bounds of the KWIK online regression algorithm \mathcal{A}_0 scale nicely to the analogous quantities in the more complicated, H -horizon RL problem.

³Strictly 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 $Q_{h+1}^* + H \sum_{l=h+1}^H (\epsilon_l + \xi_l)$.

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

- I. *The number of \perp outputted at stage h is $O\left(H \cdot \zeta_0\left(d, \frac{1}{\epsilon_0}, \frac{1}{\delta_0}\right)\right)$;*
- II. *The total number of \perp outputted during the whole run of REKWIRE in all stages is $O\left(H^2 \cdot \zeta_0\left(d, \frac{1}{\epsilon_0}, \frac{1}{\delta_0}\right)\right)$;*
- 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 $H(H - h + 1) \cdot (\epsilon_0 + \xi_0) = O(H^2 \cdot (\epsilon_0 + \xi_0))$.*

Using Corollary 1, we can prove the following theorem about the sample complexity of exploration of REKWIRE. 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 the set of features ϕ and the class of functions \mathcal{F} are sufficiently good so that $\xi_h = O(\epsilon/H^3)$. If we run $\mathcal{A}_0^{(h)}$ with $\epsilon_h = O(\epsilon/H^3)$ and $\delta_h = \delta/(2H)$ in REKWIRE, then the policy used by the agent is ϵ -optimal except in*

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

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

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

Consider any state trajectory $\rho = [s_1, s_2, \dots, s_H, s_{H+1}]$ generated by policy π_i . Two situations can occur: (i) \perp is outputted (maybe multiple times) in ρ , and (ii) \perp 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 ,

$$\begin{aligned} & Q_h^*(s_h, \pi^*(s_h, h)) - Q_h^*(s_h, \pi_i(s_h, h)) \\ & \leq Q_h^*(s, \pi^*(s_h, h)) - \hat{Q}_h(s_h, \pi_i(s_h, h)) + O(H^2(\epsilon_h + \xi_h)) \\ & = Q_h^*(s, \pi^*(s_h, h)) - \hat{Q}_h(s_h, \pi_i(s_h, h)) + O\left(\frac{\epsilon}{H}\right) \\ & \leq Q_h^*(s, \pi^*(s_h, h)) - \hat{Q}_h(s_h, \pi^*(s_h, h)) + O\left(\frac{\epsilon}{H}\right) \\ & \leq O(H^2(\epsilon_h + \xi_h)) + O\left(\frac{\epsilon}{H}\right) = O\left(\frac{\epsilon}{H}\right), \end{aligned} \tag{4}$$

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 \perp is outputted.

For any fixed start state s_1 , Lemma 2 asserts that

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

Combined with (4) 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 $p_i \leq p_0$ 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 $\mathbf{E}_{s_1 \sim \mu_0} [V^*(s_1) - V^{\pi_i}(s_1)] = O(\epsilon)$, 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 \perp is outputted at most $k = O(H^2 \cdot \zeta_0(d, H^3/\epsilon, H/\delta))$ times. If we treat the event of observing \perp in episode i as a Bernoulli trial with success probability p_i , then Lemma 3 guarantees, with probability at least $1 - \delta/2$, that we can observe all k possible \perp after

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

episodes.

Finally, we apply a union bound to two possible failures, both of which have a failure probability $\delta/2$: one is that any one copy of $\mathcal{A}_0^{(h)}$ fails, the other from the application of Lemma 3. The theorem follows by substituting $p_0 = O(\epsilon/H)$. \square

The previous theorem gives a polynomial upper bound on the number of samples needed to learn a near-optimal policy. This result does not contradict the known exponential lower bound by [17]. Their lower-bound proof involves an MDP whose value function belongs to a function class that requires exponentially many samples to learn. In the terminology of this paper, $\zeta_0(\dots)$ is exponential in H , which implies the final sample complexity is exponential in H as well.

3.3 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 [35, 40], 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 policies that maps 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 γ -discounted cumulative reward.

For discounted MDPs, since rewards are bounded and rewards in the future are exponentially down-weighted, rewards received after a certain 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

$$H \geq \frac{1}{1 - \gamma} \log \frac{1}{\epsilon(1 - \gamma)}.$$

4 Experiments

We demonstrated the practical potential of Algorithm 1 on a few benchmark problems with linear value function approximation. The algorithm of [39] was used

as the base algorithm \mathcal{A}_0 . Sarsa(0) with ϵ -greedy and Boltzmann exploration rules were used for comparison.

4.1 Problems

Figure 2 illustrates the three finite-horizon problems. CONTCOMBLOCK is a continuous version of the combination lock problem, which was designed to require a smart exploration strategy [20]. The state space $S = [0, 1]$ and the start state $s_1 = 0$. There are two actions: LEFT always resets the agent to the start state s_1 , and RIGHT takes the agent from state s to a new state $s' = s + 0.05 + \Delta$ where $\Delta \sim \mathcal{N}(0, 0.0125)$ is Gaussian noise. If the agent reaches a state $s > 0.95$, the episode terminates. Every step results in a -1 reward. Therefore, the optimal policy is to always choose RIGHT, and on average each episode takes about 19 to 20 steps to finish. We set $H = 25$. To separate the issue of exploration from that of feature selection, we used $\phi_h(s, \text{LEFT}) = Q_h(s, \text{LEFT}) \cdot [1, 0, n_1, n_2]$ and $\phi_h(s, \text{RIGHT}) = Q_h(s, \text{RIGHT}) \cdot [0, 10, n_1, n_2]$, where n_1 and n_2 were noisy values uniformly distributed in $[-0.5, 0.5]$, and $Q_h(s, a) \approx Q_h^*(s, a)$ was computed by the Bellman equation (1) on the discretized MDP. Clearly, $w_h = [1, 0.1, 0, 0]$ yields a quite accurate linear approximation to Q_h^* , and the noise was added to make the problem less trivial.

MOUNTAINCAR involves driving a car to the hilltop. It has two continuous state variables and three actions. Every step results in a -1 reward and the state transition is governed by a system of nonlinear equations. The start state is $s_1 = [0, 0]$ and we set $H = 75$. As before, $\phi_h(s, a) = Q_h(s, a) \cdot [1, n_1, n_2]$, and hence $w_h = [1, 0, 0]$ yields a quite accurate linear approximation to Q_h^* .

EXPRESSWORLD is a variant of PUDDLEWORLD. In this problem, there is a two-dimensional continuous grid world ($S = [0, 1]^2$) and an agent can move along four directions (NORTH, EAST, SOUTH, and WEST) to reach the goal region in the north-east corner while trying to avoid two puddles (the shaded region in the figure). Each step results in a -1 reward plus penalty for entering the puddles. To make exploration harder, the agent starts in $s_1 = [x, 0]$ where $x \sim \text{Uniform}[0, 1]$, and an “express lane” of 0.15 wide is added so that the immediate reward is -0.5 instead of -1 if the agent moves inside this lane. Hence, the agent has to learn how to avoid puddles, as well as to explore actively to discover the goal and the express lane. We set $H = 45$. As

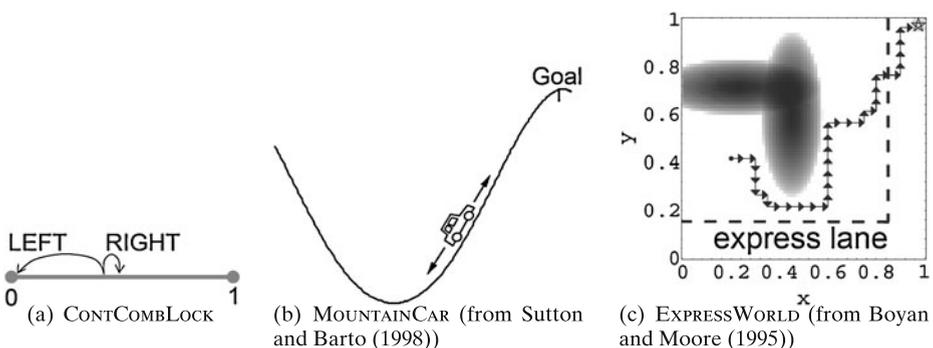
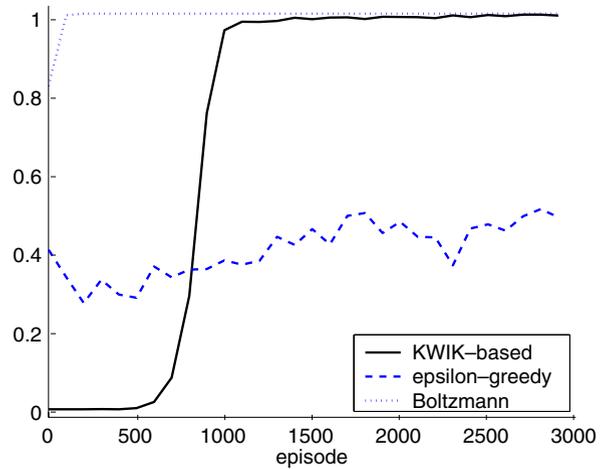
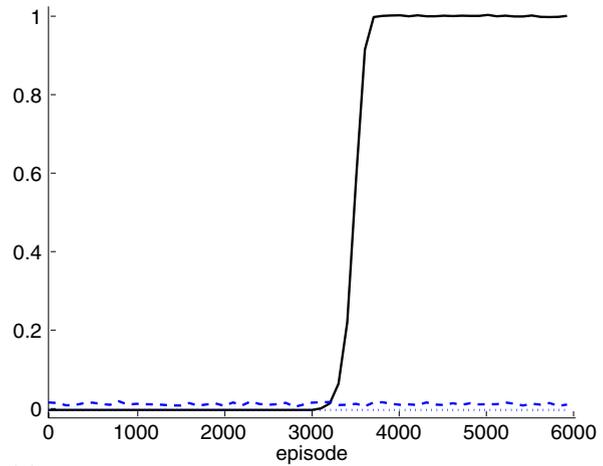


Fig. 2 Problems used in experiments

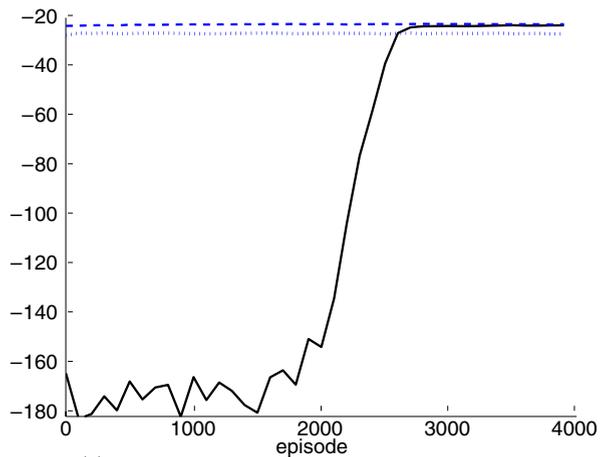
Fig. 3 Comparisons of REKWIRE to Sarsa(0) with ϵ -greedy and Boltzmann exploration rules. All results are averaged over 20 runs



(a) Probabilities of reaching the goal states in CONTCOMBLOCK

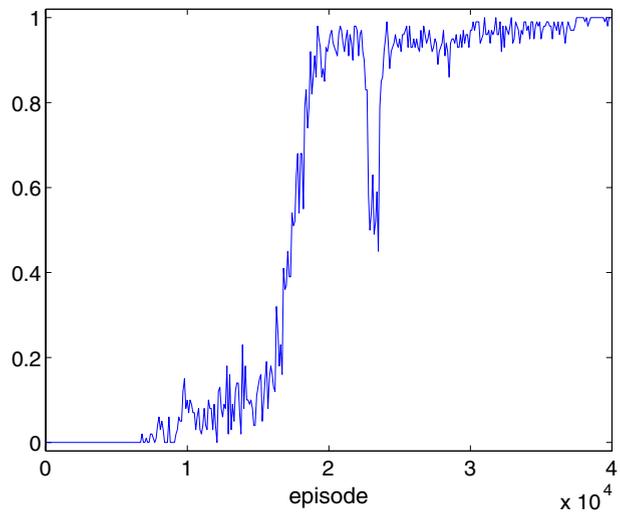


(b) Probabilities of reaching the goal states in MOUNTAINCAR



(c) Per-episode total reward in PUDDLEWORLD

Fig. 4 Probability of reaching the goal within $H = 25$ steps in `CONTCOMBLOCK` when RBF features were used. The results were computed based on a typical run and the probabilities were computed using every other block of 100 episodes



before, $\phi_h(s, a) = \hat{Q}_h(s, a) \cdot [1, n_1, n_2]$, and hence $w_h = [1, 0, 0]$ yields a quite accurate linear approximation to Q_h^* .

4.2 Results

Figure 3 summarizes the results on all three problems. We picked parameters that seemed to work best for each algorithm. Since the number of steps to the goal states is very close to H in `CONTCOMBLOCK` and `MOUNTAINCAR`, we instead plotted the probabilities of reaching the goal within H steps. In contrast, we evaluated per-episode reward in `EXPRESSWORLD` as it is necessary to distinguish whether the agent discovers the express lane or not.

It is observed that our algorithm consistently solved all three problems, while ϵ -greedy and Boltzmann rules worked satisfactorily for some and failed for others. For example, ϵ -greedy was inefficient in `CONTCOMBLOCK`, as expected; the Boltzmann rule failed to converge to the optimal policy in `EXPRESSWORLD` although it quickly learned a good policy early on. Our algorithm, however, seemed too conservative and converged more slowly. A partial explanation is that it learned H weight vectors, while Sarsa learned only one weight vector and used it to compute a policy in all stages. It remains an interesting open question how we may avoid representing the value function using H weights in `REKWIRE` and this is particularly important when it is applied to discounted problems.

We next take a closer look at `REKWIRE` in `CONTCOMBLOCK`, whose value functions can be easily visualized. To make the problem more interesting, we ran `REKWIRE` with RBF features with 4 RBF centers located evenly in the state space $[0, 1]$. Figure 4 plots the probability of reaching the goal within $H = 25$ steps for our algorithm, which is consistently high after around 25,000 episodes.⁴ In contrast, both ϵ -greedy and Boltzmann rules failed this task in our experiments.

⁴Since the feature vector has 10 dimensions (1 constant feature plus 4 RBF centers per action) and $H = 25$, each parameter required about 1000 data to estimate.

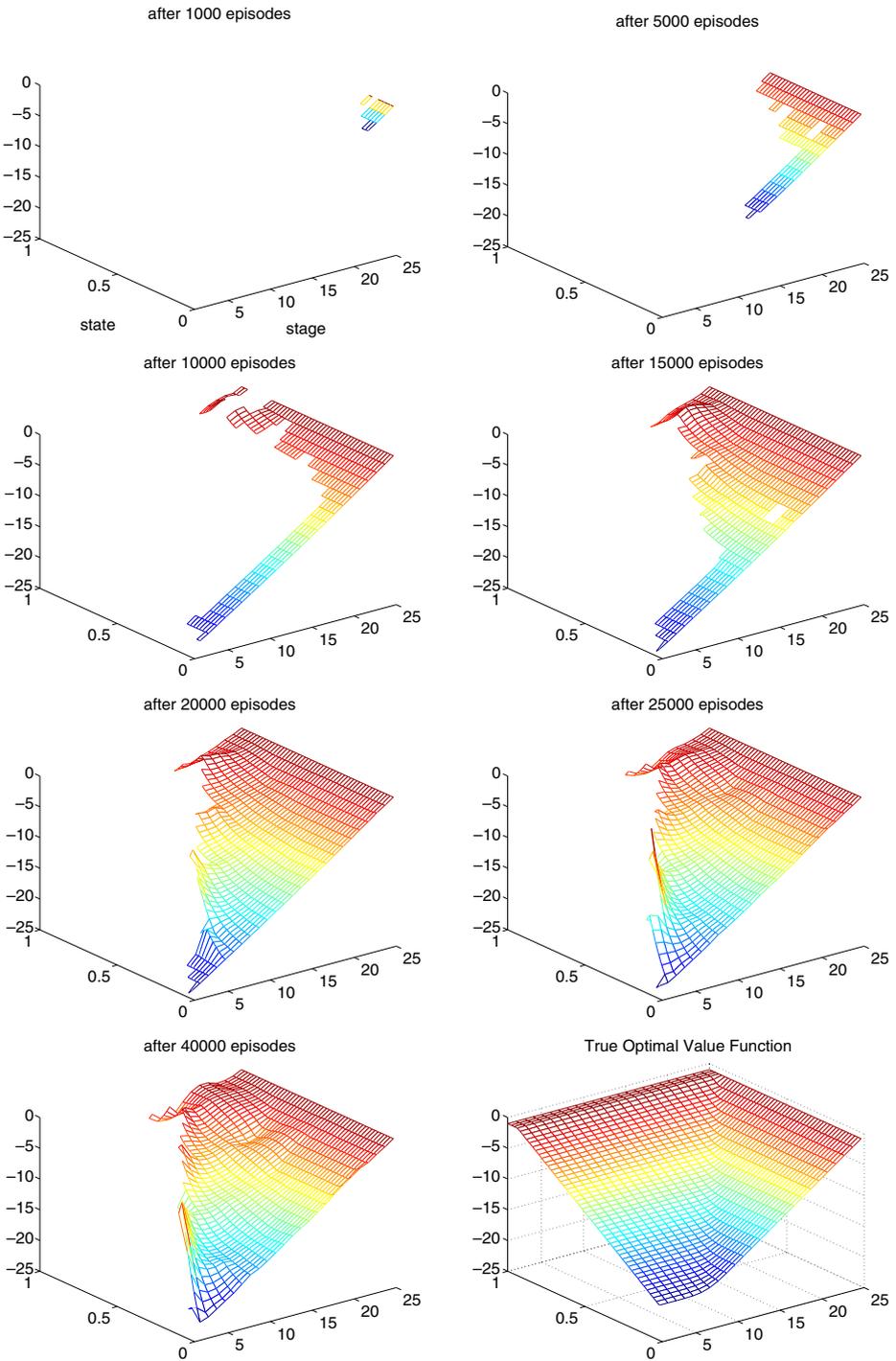


Fig. 5 Value function evolution when RBF features are used. The *bottom-right* plot gives the true V^* function

Figure 5 gives a few snapshots of the value function $\hat{V}_h(s) = \max_a \hat{Q}_h(s, a)$ computed by REKWIRE: $\hat{V}_h(s, a)$ is a valid prediction only when $\hat{Q}_h(s, a)$ are valid for both actions; otherwise, $\hat{V}_h(s, a) = \perp$ and the value is not plotted in the figure. At the beginning of learning, most states' values are unknown. The algorithm has to work backwards by learning Q_h^* from larger h to smaller h , as shown in the plots. The plots show that the “known” region essentially stopped growing after about 25,000 episodes, and the policy stabilized after that, which is consistent with the learning curve in Fig. 4. Comparing to the true value function, we found that the value function computed by REKWIRE was indeed very close to the true value for “known” regions, exactly as we would expect.

5 Related work

This work is a generalization of our earlier result for linear value functions [29], which is most relevant to the original KWIK online linear regression framework proposed by [39]. The only difference in the problem formulations 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 .

Although a few algorithms with provably efficient exploration exist for finite MDPs (see [36] for a survey), it remains a challenging problem when function approximation is used. A few variants of E^3 and Rmax approximate an MDP model compactly, including metric E^3 [16] and KWIK-Rmax [32]. A common subroutine in these model-based algorithms is solving an MDP model, which is computationally expensive [8, 17]. In contrast, the *model-free* approach taken in this paper avoids this problem completely by learning the value function directly. With a learned value function, finding the greedy action takes at most $O(|A| \tau_0)$ time per step. Other function-approximation-based methods, such as variants of Rmax [14, 30] and temporal difference learning [10, 13], however, lack theoretical guarantees in general.

Our algorithm shares some similarity with a few existing algorithms. The Grow-Support algorithm [4] also uses value-function estimates for backups when the estimates are “trusted”. But Grow-Support assumes complete knowledge about the MDP model and thus focuses on stability issues when combining function approximation and dynamic programming. In contrast, REKWIRE is a learning algorithm and has to explore in the MDP to gather information needed for computing an optimal policy. Another similar, bottom-up algorithm is proposed by [12] in the same H -horizon MDP setting. The algorithm is model-based, learning MDP dynamics in every stage, and is for finite MDP only. REKWIRE, on the other hand, is model-free and is not limited to finite MDPs.

Finally, we compare our work to existing reduction works in RL. Reductions from RL to classification based on LSPI (e.g., [11, 23], and [28]) normally assume a set of samples are given, and it is not obvious how efficient exploration is done if these algorithms are run online. The reduction by [25] builds classifiers backwards from stage H to stage 1, similar to REKWIRE. Their analysis reveals a connection between

classification error and policy loss, but their reduction has to query a generative model for exponentially (in H) many times in order to achieve near-optimality of the learned policy [24]. In contrast, our reduction is efficient and can be run in an entirely online fashion without assuming a generative model. Another reduction to classification with performance guarantee is by [3]. They assume knowledge of state distribution in each stage when a (near-)optimal policy is being followed, and then use this distribution to the classifiers in that stage. In contrast, we do not require such strong prior knowledge.

6 Discussions and conclusions

In this work, we have presented a theoretically sound connection between KWIK online regression and reinforcement learning, suggesting an alternative way to tackle the general RL problem. Specifically, we prove approximation error bound and sample complexity of exploration bounds for our reduction algorithm, REKWIRE. A set of proof-of-concept experiments are also provided to demonstrate the potential value of this algorithm.

We may compare the sample complexity bound of REKWIRE to previous results for finite MDPs. For the purpose of comparison, we may define \mathcal{F} as the set of all possible Q-value tables that maps (finitely many) states to actions. In this case, the feature ϕ is sometimes called an indicator feature since it has a dimension of $d = |S| |A|$, with one component for each state–action pair. Applying Hoeffding’s inequality with the “input-partition” algorithm of [31] yields $\zeta_0(d, \epsilon, \delta) = \tilde{O}(d/\epsilon^2)$. Here, the notation $\tilde{O}(\cdot)$ is like the big-O notation but suppresses dependence on logarithmic factors. The resulting sample complexity bound of exploration for REKWIRE (c.f., Theorem 2) has a better dependence on $1/\epsilon$, but worse on H , when compared to the model-free delayed Q-learning [37]. However, the bound is strictly worse than the best available bound for Rmax [15]. It is interesting to design a more efficient reduction or to obtain a tightened analysis for REKWIRE in the future.

Second, it is important to study special function classes for which admissible algorithms may be designed. It is known that KWIK is a harder-to-learning model than the Probably Approximate Correct and the Mistake Bound models [31]. In some cases, admissible algorithms may not exist if we allow the input vectors to be generated by an adversary [29]. However, such a worst-case scenario may not happen in most reinforcement-learning problems encountered in practice. Common assumptions like small mixing time or Lipschitz continuity may be necessary for devising an admissible algorithm.

Third, given the hardness result in the KWIK model, an important direction is to investigate alternative models that are easier to work with and allow similar performance guarantees we achieve here.

Finally, it is interesting to find a more efficient reduction for discounted RL that do not involve an intermediate conversion to an H -horizon problem, as described in Section 3.3. Such a direction reduction may lead to improved sample complexity of exploration bounds.

Acknowledgements We thank Alex Strehl for helpful discussions and the anonymous reviewers for suggestions that have significantly improved an earlier version of the manuscript. The work is primarily supported by NSF-ITR-0325281 and DARPA’s Transfer Learning Program.

Appendix: Lemmas

The appendix gives three lemmas used in our analysis in Section 3.2. Slightly different forms of these lemmas have appeared in the literature. We give full proofs here for completeness.

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 $i = 1, 2$. Then, $|\max_{x \in X} f_1(x) - \max_{x \in X} f_2(x)| \leq \max_{x \in X} |f_1(x) - f_2(x)|$.*

Proof Define $x_i = \arg \max_{x \in X} f_i(x)$ for $i = 1, 2$, and then $\max_{x \in X} f_1(x) - \max_{x \in X} f_2(x) = f_1(x_1) - f_2(x_2)$. On the one hand, we have

$$f_1(x_1) - f_2(x_2) \geq f_1(x_2) - f_2(x_2) \geq -\max_{x \in X} |f_1(x) - f_2(x)|;$$

on the other hand, we have

$$f_1(x_1) - f_2(x_2) \leq f_1(x_1) - f_2(x_1) \leq \max_{x \in X} |f_1(x) - f_2(x)|.$$

Hence, $|f_1(x_1) - f_2(x_2)| \leq \max_{x \in X} |f_1(x) - f_2(x)|$, and the lemma follows. □

Lemma 2 (From [26]) *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,*

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

where \mathbf{E}_π stands for the expectation with respect to the probability distributions of trajectories $\rho = [s_1, s_2, \dots, s_H, s_{H+1}]$ 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,

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

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

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

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

The last lemma follows from the standard multiplicative form of the inequality of [7]. The proof is from [27].

Lemma 3 *Let $x_1, x_2, x_3, \dots \in \{0, 1\}$ be a sequence of m independent Bernoulli trials, each with a success probability at least μ : $\mathbf{E}[x_i] \geq \mu$, for some constant $\mu > 0$. Then for any $k \in \mathbb{N}$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$, $x_1 + x_2 + \dots + x_m \geq k$ if*

$$m \geq \frac{2}{\mu} \left(k + \ln \frac{1}{\delta} \right). \tag{5}$$

Proof The multiplicative form of Chernoff’s inequality (see, e.g., [19]) states that

$$\Pr \left(\sum_{i=1}^m x_i < k \right) \leq \exp \left(-\frac{m\mu\alpha^2}{2} \right),$$

where $k = (1 - \alpha)m\mu$, and so $\alpha = 1 - \frac{k}{m\mu}$. To guarantee the failure probability on the left-hand side is less than δ , it suffices to set m so that the right-hand side is no greater than δ , yielding

$$m \geq \frac{2}{\mu\alpha^2} \ln \frac{1}{\delta} = \frac{2 \ln \frac{1}{\delta}}{\mu \left(1 - \frac{k}{m\mu} \right)^2}. \tag{6}$$

We want to find a large enough value for m to satisfy the inequality above. Using simple algebra, the inequality above can be rewritten as

$$m - \sqrt{\frac{2m}{\mu} \ln \frac{1}{\delta}} - \frac{k}{\mu} \geq 0.$$

Solving this quadratic equation for the unknown value \sqrt{m} gives

$$\sqrt{m} \geq \sqrt{\frac{1}{2\mu} \ln \frac{1}{\delta}} + \sqrt{\frac{1}{2\mu} \ln \frac{1}{\delta} + \frac{k}{\mu}}.$$

Finally, using the elementary inequality, $x + y \leq \sqrt{2(x^2 + y^2)}$, we find the value of m given in (5) suffices to guarantee (6). \square

References

1. Asmuth, J., Li, L., Littman, M. L., Nouri, A., Wingate, D.: A Bayesian sampling approach to exploration in reinforcement learning. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-09), pp. 19–26 (2009)
2. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.* **3**, 397–422 (2002)
3. Bagnell, J.A., Kakade, S., Ng, A.Y., Schneider, J.: Policy search by dynamic programming. *Adv. Neural Inf. Process. Syst.* **16** (NIPS-03), 831–838 (2004)
4. Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: safely approximating the value function. *Adv. Neural Inf. Process. Syst.* **7**, 369–376 (1995)
5. Brafman, R.I., Tenenbholz, M.: R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.* **3**, 213–231 (2002)
6. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press (2006)
7. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Stat.* **23**, 493–507 (1952)

8. Chow, C.-S., Tsitsiklis, J.N.: The complexity of dynamic programming. *J. Complex.* **5**, 466–488 (1989)
9. Duff, M.O.: *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes*. Doctoral dissertation, University of Massachusetts, Amherst, MA (2002)
10. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with Gaussian processes. In: *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML-05)*, pp. 201–208 (2005)
11. Fern, A., Yoon, S.W., Givan, R.: Approximate policy iteration with a policy language bias: solving relational Markov decision processes. *J. Artif. Intell. Res.* **25**, 75–118 (2006)
12. Fiechter, C.-N.: Efficient reinforcement learning. In: *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pp. 88–97 (1994)
13. Geist, M., Pietquin, O., Fricout, G.: Kalman temporal differences: the deterministic case. In: *Proceedings of the IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-09)*, pp. 185–192 (2009)
14. Jong, N.K., Stone, P.: Model-based exploration in continuous state spaces. In: *Proceedings of the Seventh International Symposium on Abstraction, Reformulation and Approximation (SARA-07)*, pp. 258–272 (2007)
15. Kakade, S.: *On the Sample Complexity of Reinforcement Learning*. Doctoral dissertation, University College London, UK (2003)
16. Kakade, S., Kearns, M.J., Langford, J.: Exploration in metric state spaces. In: *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 306–312 (2003)
17. Kearns, M.J., Mansour, Y., Ng, A.Y.: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Mach. Learn.* **49**, 193–208 (2002)
18. Kearns, M.J., Singh, S.P.: Near-optimal reinforcement learning in polynomial time. *Mach. Learn.* **49**, 209–232 (2002)
19. Kearns, M.J., Vazirani, U.V.: *An Introduction to Computational Learning Theory*. MIT Press (1994)
20. Koenig, S., Simmons, R.G.: The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Mach. Learn.* **22**, 227–250 (1996)
21. Kolter, J.Z., Ng, A.Y.: Near Bayesian exploration in polynomial time. In: *Proceedings of the Twenty-Sixth International Conference on Machine Learning (ICML-09)*, pp. 513–520 (2009)
22. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. *J. Mach. Learn. Res.* **4**, 1107–1149 (2003)
23. Lagoudakis, M.G., Parr, R.: Reinforcement learning as classification: leveraging modern classifiers. In: *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, pp. 424–431 (2003)
24. Langford, J., Zadrozny, B.: Reducing T -step reinforcement learning to classification. In: *Proceedings of the Machine Learning Reductions Workshop*. Chicago, IL (2003)
25. Langford, J., Zadrozny, B.: Relating reinforcement learning performance to classification performance. In: *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML-05)*, pp. 473–480 (2005)
26. Li, L.: *Focus of Attention in Reinforcement Learning*. Master's thesis, University of Alberta, Edmonton, AB, Canada (2004)
27. Li, L.: *A Unifying Framework for Computational Reinforcement Learning Theory*. Doctoral dissertation, Rutgers University, New Brunswick, NJ (2009)
28. Li, L., Bulitko, V., Greiner, R.: Focus of attention in reinforcement learning. *J. Univers. Comput. Sci.* **13**, 1246–1269 (2007)
29. Li, L., Littman, M.L.: Efficient value-function approximation via online linear regression. In: *Proceedings of the Tenth International Symposium on Artificial Intelligence and Mathematics (AMAI-08)* (2008)
30. Li, L., Littman, M.L., Mansley, C.R.: Online exploration in least-squares policy iteration. In: *Proceedings of the Eighteenth International Conference on Agents and Multiagent Systems (AAMAS-09)* (2009)
31. Li, L., Littman, M.L., Walsh, T.J.: Knows what it knows: A framework for self-aware learning. In: *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML-08)*, pp. 568–575 (2008)
32. Li, L., Littman, M.L., Walsh, T.J., Strehl, A.L.: Knows what it knows: a framework for self-aware learning. (2010, in submission)
33. Ortner, R., Auer, P., Jaksch, T.: Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.* **11**, 1563–1600 (2010)

34. Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete Bayesian reinforcement learning. In: Proceedings of the Twenty-Third International Conference on Machine Learning (ICML-06), pp. 697–704 (2006)
35. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience, New York (1994)
36. Strehl, A.L., Li, L., Littman, M.L.: Reinforcement learning in finite MDPs: PAC analysis. *J. Mach. Learn. Res.* **10**, 2413–2444 (2009)
37. Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC model-free reinforcement learning. In: Proceedings of the Twenty-Third International Conference on Machine Learning, pp. 881–888 (2006)
38. Strehl, A.L., Littman, M.L.: A theoretical analysis of model-based interval estimation. In: Proceedings of the Twenty-Second Conference on Machine Learning, pp. 857–864 (2005)
39. Strehl, A.L., Littman, M.L.: Online linear regression and its application to model-based reinforcement learning. *Adv. Neural Inf. Process. Syst.* **20**, 1417–1424 (2008)
40. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
41. Thrun, S.: The role of exploration in learning control. In: White, D.A., Sofge, D.A. (eds.) *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pp. 527–559. Van Nostrand Reinhold (1992)