PAC Model-Free Reinforcement Learning

Alexander L. Strehl STREHL@CS.RUTGERS.EDU
Lihong Li LIHONG@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, Piscataway, NJ 08854 USA

Eric Wiewiora EWIEWIOR@CS.UCSD.EDU
Computer Science and Engineering Department University of California, San Diego

John Langford JL@HUNCH.NET
TTI-Chicago, 1427 E 60th Street, Chicago, IL 60637 USA

Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, Piscataway, NJ 08854 USA

Abstract

For a Markov Decision Process with finite state (size S) and action spaces (size A per state), we propose a new algorithm—Delayed Q-Learning. We prove it is PAC, achieving near optimal performance except for $\tilde{O}(SA)$ timesteps using $O(SA)$ space, improving on the $\tilde{O}(S^2A)$ bounds of best previous algorithms. This result proves efficient reinforcement learning is possible without learning a model of the MDP from experience. Learning takes place from a single continuous thread of experience—no resets nor parallel sampling is used. Beyond its smaller storage and experience requirements, Delayed Q-learning's per-experience computation cost is much less than that of previous PAC algorithms.

1. Introduction

In the reinforcement-learning (RL) problem (Sutton & Barto, 1998), an agent acts in an unknown or incompletely known environment with the goal of maximizing an external reward signal. One of the fundamental obstacles in RL is the exploration-exploitation dilemma: whether to act to gain new information (explore) or to act consistently with past experience to maximize reward (exploit). This paper models the RL problem as a Markov Decision Process (MDP) environment with finite state and action spaces.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

When evaluating RL algorithms, there are three essential traits to consider: *space complexity*, *computational complexity*, and *sample complexity*. We define a *timestep* to be a single interaction with the environment. Space complexity measures the amount of memory required to implement the algorithm while computational complexity measures the amount of operations needed to execute the algorithm, per timestep. Sample complexity measures the amount of timesteps for which the algorithm does not behave near optimally or, in other words, the amount of experience it takes to learn to behave well.

We will call algorithms whose sample complexity can be bounded by a polynomial in the environment size and approximation parameters, with high probability, *PAC-MDP* (*Probably Approximately Correct in Markov Decision Processes*). All algorithms known to be PAC-MDP to date involve the maintenance and solution (often by value iteration or mathematical programming) of an internal MDP model. Such algorithms, including $R_{\max}$ (Brafman & Tennenholtz, 2002), E^3 (Kearns & Singh, 2002), and MBIE (Strehl & Littman, 2005), are called *model-based algorithms* and have relatively high space and computational complexities. Another class of algorithms, including most forms of Q-learning (Watkins & Dayan, 1992), make no effort to learn a model and can be called *model free*.

It is difficult to articulate a hard and fast rule dividing model-free and model-based algorithms, but model-based algorithms generally retain some transition information during learning whereas model-free algorithms only keep value-function information. Instead of formalizing this intuition, we have decided to adopt a crisp, if somewhat unintuitive, definition. For our purposes, a *model-free RL algorithm* is one whose space complexity is asymptotically less than the space required to store an MDP.

Definition 1 *A learning algorithm is said to be* **model free** *if its space complexity is always $o(S^2A)$, where S is the number of states and A is the number of actions of the MDP used for learning.*

Although they tend to have low space and computational complexity, no model-free algorithm has been proven to be PAC-MDP. In this paper, we present a new model-free algorithm, **Delayed Q-learning**, and prove it is the first such algorithm.

The hardness of learning an arbitrary MDP as measured by sample complexity is still relatively unexplored. For simplicity, we let $\tilde{O}(\cdot)$ ($\tilde{\Omega}(\cdot)$) represent $O(\cdot)$ ($\Omega(\cdot)$) where logarithmic factors are ignored. When we consider only the dependence on S and A, the lower bound of Kakade (2003) says that with probability greater than $1-\delta$, the sample complexity of any algorithm will be $\tilde{\Omega}(SA)$. However, the best upper bound known provides an algorithm whose sample complexity is $\tilde{O}(S^2A)$ with probability at least $1-\delta$. In other words, there are algorithms whose sample complexity is known to be no greater than approximately the number of bits required to specify an MDP to fixed precision. However, there has been no argument proving that learning to act near-optimally takes as long as approximating the dynamics of an MDP. We solve this open problem, first posed by Kakade (2003), by showing that Delayed Q-learning has sample complexity $\tilde{O}(SA)$, with high probability. Our result therefore proves that efficient RL is possible without learning a model of the environment from experience.

2. Definitions and Notation

This section introduces the Markov Decision Process notation used throughout the paper; see Sutton and Barto (1998) for an introduction. An MDP M is a five tuple $\langle S, A, T, R, \gamma \rangle$, where S is the state space, A is the action space, $T : S \times A \times S \rightarrow \mathbb{R}$ is a transition function, $R : S \times A \rightarrow \mathbb{R}$ is a reward function, and $0 \leq \gamma < 1$ is a discount factor on the summed sequence of rewards. We also let S and A denote the number of states and the number of actions, respectively. From state s under action a, the agent receives a random reward r, which has expectation $R(s,a)$, and is transported to state s' with probability $T(s'|s,a)$. A policy is a strategy for choosing actions. Only deterministic policies are dealt with in this paper. A stationary policy is one that produces an action based on only the current state. We assume that rewards all lie between 0 and 1. For any policy π, let $V_M^\pi(s)$ ($Q_M^\pi(s,a)$) denote the discounted, infinite-horizon value (action-value or Q-value) function for π in M (which may be omitted from the notation) from state s. If T is a positive integer, let $V_M^\pi(s,T)$ denote the T-step value function of policy π. Specifically, $V_M^\pi(s) = E[\sum_{j=1}^\infty \gamma^{j-1} r_j]$ and $V_M^\pi(s,T) = E[\sum_{j=1}^T \gamma^{j-1} r_j]$ where $[r_1, r_2, \ldots]$ is the reward sequence generated by following policy π from state s. These expectations are taken over all possible infinite paths the agent might follow. The optimal policy is denoted π^* and has value functions $V_M^*(s)$ and $Q_M^*(s,a)$. Note that a policy cannot have a value greater than $1/(1-\gamma)$ in any state.

3. Learning Efficiently

In our discussion, we assume that the learner receives S, A, ϵ, δ, and γ as input. The learning problem is defined as follows. The agent always occupies a single state s of the MDP M. The learning algorithm is told this state and must select an action a. The agent receives a reward r and is then transported to another state s' according to the rules from Section 2. This procedure then repeats forever. The first state occupied by the agent may be chosen arbitrarily.

There has been much discussion in the RL community over what defines efficient learning or how to define sample complexity. For any fixed ϵ, Kakade (2003) defines the **sample complexity of exploration** (sample complexity, for short) of an 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[1], s_t at time t (formally $V^{\mathcal{A}_t}(s_t) < V^*(s_t) - \epsilon$). We believe this definition captures the essence of measuring learning. An algorithm $\mathcal{A}$ is then said to be **PAC-MDP** (Probably Approximately Correct in Markov Decision Processes) if, for any ϵ and δ, the sample complexity of $\mathcal{A}$ is less than some polynomial in the relevant quantities $(S, A, 1/\epsilon, 1/\delta, 1/(1-\gamma))$, with probability at least $1-\delta$.

The above definition penalizes the learner for executing a non-ϵ-optimal policy rather than for a non-optimal policy. Keep in mind that, with only a finite amount of experience, no algorithm can identify the optimal policy with complete confidence. In addition, due to noise, any algorithm may be misled about the underlying dynamics of the system. Thus, a failure probability of at most δ is allowed. See Kakade (2003) for a full motivation of this performance measure.

[1] Note that $\mathcal{A}_t$ is completely defined by $\mathcal{A}$ and the agent's history up to time t.

4. Delayed Q-learning

In this section we describe a new reinforcement-learning algorithm, Delayed Q-learning.

Delayed Q-learning maintains Q-value estimates, $Q(s, a)$ for each state-action pair (s, a). At time $t(= 1, 2, \ldots)$, let $Q_t(s, a)$ denote the algorithm's current Q-value estimate for (s, a) and let $V_t(s)$ denote $\max_{a \in A} Q_t(s, a)$. The learner always acts greedily with respect to its estimates, meaning that if s is the tth state reached, $a' := \operatorname{argmax}_{a \in A} Q_t(s, a)$ is the next action chosen.

In addition to Q-value estimates, the algorithm maintains a Boolean flag $\texttt{LEARN}(s, a)$, for each (s, a). Let $\texttt{LEARN}_t(s, a)$ denote the value of $\texttt{LEARN}(s, a)$ at time t, that is, the value immediately before the tth action is taken. The flag indicates whether the learner is considering a modification to its Q-value estimate $Q(s, a)$. The algorithm also relies on two free parameters, $\epsilon_1 \in (0, 1)$ and a positive integer m. In the analysis of Section 5, we provide precise values for these parameters in terms of the other inputs (S, A, ϵ, δ, and γ) that guarantee the resulting algorithm is PAC-MDP. Finally, a counter $l(s, a)$ ($l_t(s, a)$ at time t) is also maintained for each (s, a). Its value represents the amount of data (sample points) acquired for use in an upcoming update of $Q(s, a)$. Once m samples are obtained and $\texttt{LEARN}(s, a)$ is $\texttt{true}$, an update is attempted.

4.1. Initialization of the Algorithm

The Q-value estimates are initialized to $1/(1 - \gamma)$, the counters $l(s, a)$ to zero, and the $\texttt{LEARN}$ flags to $\texttt{true}$. That is, $Q_1(s, a) = 1/(1 - \gamma)$, $l_1(s, a) = 0$, and $\texttt{LEARN}_1(s, a) = \texttt{true}$ for all $(s, a) \in S \times A$.

4.2. The Update Rule

Suppose that at time $t \geq 1$, action a is performed from state s, resulting in an *attempted update*, according to the rules to be defined in Section 4.3. Let $s_{k_1}, s_{k_2}, \ldots, s_{k_m}$ be the m most recent next-states observed from executing (s, a), at times $k_1 < k_2 < \cdots < k_m$, respectively ($k_m = t$). For the remainder of the paper, we also let r_i denote the ith reward received during execution of Delayed Q-learning.

Thus, at time k_i, action a was taken from state s, resulting in a transition to state s_{k_i} and an immediate reward r_{k_i}. After the tth action, the following update occurs:

$$Q_{t+1}(s, a) = \frac{1}{m} \sum_{i=1}^{m} (r_{k_i} + \gamma V_{k_i}(s_{k_i})) + \epsilon_1 \qquad (1)$$

as long as performing the update would result in a new Q-value estimate that is at least ϵ_1 smaller than the previous estimate. In other words, the following equation must be satisfied for an update to occur:

$$Q_t(s, a) - \left(\frac{1}{m} \sum_{i=1}^{m} (r_{k_i} + \gamma V_{k_i}(s_{k_i})) \right) \geq 2\epsilon_1. \qquad (2)$$

If any of the above conditions do not hold, then no update is performed. In this case, $Q_{t+1}(s, a) = Q_t(s, a)$.

4.3. Maintenance of the LEARN Flags

We provide an intuition behind the behavior of the $\texttt{LEARN}$ flags. Please see Section 4.4 for a formal description of the update rules. The main computation of the algorithm is that every time a state-action pair (s, a) is experienced m times, an update of $Q(s, a)$ is attempted as in Section 4.2. For our analysis to hold, however, we cannot allow an infinite number of attempted updates. Therefore, attempted updates are only allowed for (s, a) when $\texttt{LEARN}(s, a)$ is $\texttt{true}$. Besides being set to $\texttt{true}$ initially, $\texttt{LEARN}(s, a)$ is also set to $\texttt{true}$ when any state-action pair is updated (because our estimate $Q(s, a)$ may need to reflect this change). $\texttt{LEARN}(s, a)$ can only change from $\texttt{true}$ to $\texttt{false}$ when no updates are made during a length of time for which (s, a) is experienced m times and the next attempted update of (s, a) fails. In this case, no more attempted updates of (s, a) are allowed until another Q-value estimate is updated.

4.4. Implementation of Delayed Q-learning

We provide an efficient implementation, Algorithm 1, of Delayed Q-learning that achieves our desired computational and space complexities.

4.5. Discussion

Delayed Q-learning is similar in many aspects to traditional Q-learning. Suppose that at time t, action a is taken from state s resulting in reward r_t and next-state s'. Then, the Q-learning update is

$$Q_{t+1}(s, a) = (1 - \alpha_t) Q_t(s, a) + \alpha_t (r_t + \gamma V_t(s')) \quad (3)$$

where $\alpha_t \in [0, 1]$ is the learning rate. Note that if we let $\alpha_t = 1/(l_t(s, a) + 1)$, then m repetitions of Equation 3 is similar to the update for Delayed Q-learning (Equation 1) minus a small bonus of ϵ_1. However, Q-learning changes its Q-value estimates on every timestep, while Delayed Q-learning waits for m sample updates to make any changes. This variation has an averaging effect that mitigates some of the effects of randomness and, when combined with the bonus of

1 Delayed Q-learning

```
Inputs: γ, S, A, m, ε₁
for all (s, a) do
    Q(s, a) ← 1/(1 − γ)    // Q-value estimates
    U(s, a) ← 0    // used for attempted updates
    l(s, a) ← 0    // counters
    t(s, a) ← 0    // time of last attempted update
    LEARN(s, a) ← true    // the LEARN flags
end for
t* ← 0    // time of most recent Q-value change
for t = 1, 2, 3, ··· do
    Let s denote the state at time t.
    Choose action a := argmax_{a'∈A} Q(s, a').
    Let r be the immediate reward and s' the next
    state after executing action a from state s.
    if LEARN(s, a) = true then
        U(s, a) ← U(s, a) + r + γ max_{a'} Q(s', a')
        l(s, a) ← l(s, a) + 1
        if l(s, a) = m then
            if Q(s, a) − U(s, a)/m ≥ 2ε₁ then
                Q(s, a) ← U(s, a)/m + ε₁
                t* ← t
            else if t(s, a) ≥ t* then
                LEARN(s, a) ← false
            end if
            t(s, a) ← t, U(s, a) ← 0, l(s, a) ← 0
        end if
    else if t(s, a) < t* then
        LEARN(s, a) ← true
    end if
end for
```

ϵ_1, achieves optimism ($Q(s,a) \geq Q^*(s,a)$) with high probability (see Lemma 2).

The property of optimism is useful for safe exploration and appears in many existing RL algorithms. The intuition is that if an action's Q-value is overly optimistic the agent will learn much by executing that action. Since the action-selection strategy is greedy, the Delayed Q-learning agent will tend to choose overly optimistic actions, therefore achieving directed exploration when necessary. If sufficient learning has been completed and all Q-values are close to their true Q^*-values, selecting the maximum will guarantee near-optimal behavior. In the next section, we provide a formal argument that Delayed Q-learning exhibits sufficient exploration for learning, specifically that it is PAC-MDP.

5. Analysis

We briefly address space and computational complexity before focusing on analyzing the sample complexity of Delayed Q-learning.

5.1. Space and Computational Complexity

An implementation of Delayed Q-learning, as in Section 4.4, can be achieved with $O(SA)$ space complexity[2]. With use of a priority queue for choosing actions with maximum value, the algorithm can achieve $O(\ln A)$ computational complexity per timestep. Asymptotically, Delayed Q-learning's computational and space complexity are on par with those of Q-learning. In contrast, the $R_{\max}$ algorithm, a standard model-based method, has worst-case space complexity of $\Theta(S^2 A)$ and computational complexity of $\Omega(S^2 A)$ per experience.

5.2. Sample Complexity

The main result of this section, whose proof is provided in Section 5.2.1, is that the Delayed Q-learning algorithm is PAC-MDP:

Theorem 1 *Let M be any MDP and let ϵ and δ be two positive real numbers. If Delayed Q-learning is executed on MDP M, then it will follow an ϵ-optimal policy on all but $O\left(\frac{SA}{(1-\gamma)^8\epsilon^4}\ln\frac{1}{\delta}\ln\frac{1}{\epsilon(1-\gamma)}\ln\frac{SA}{\delta\epsilon(1-\gamma)}\right)$ timesteps, with probability at least $1-\delta$.*

To analyze the sample complexity of Delayed Q-learning, we first bound the number of successful updates. By Condition 2, there can be no more than

$$\kappa := \frac{1}{(1-\gamma)\epsilon_1} \tag{4}$$

successful updates of a fixed state-action pair (s,a). This bound follows from the fact that $Q(s,a)$ is initialized to $1/(1-\gamma)$ and that every successful update of $Q(s,a)$ results in a decrease of at least ϵ_1. Also, by our assumption of non-negative rewards, it is impossible for any update to result in a negative Q-value estimate. Thus, the total number of successful updates is at most $SA\kappa$.

Now, consider the number of attempted updates for a single state-action pair (s,a). At the beginning of learning, `LEARN`$(s,a) =$ `true`, which means that once (s,a) has been experienced m times, an attempted update will occur. After that, a successful update of some

[2]We measure complexity assuming individual numbers require unit storage and can be manipulated arithmetically in unit time. Removing this assumption increases space and computational complexities by logarithmic factors.

Q-value estimate must take place for $\texttt{LEARN}(s,a)$ to be set to `true`. Therefore, there can be at most $1+SA\kappa$ attempted updates of (s,a). Hence, there are at most

$$SA(1+SA\kappa) \tag{5}$$

total attempted updates.

During timestep t of learning, we define K_t to be the set of all state-action pairs (s,a) such that:

$$Q_t(s,a) - \left(R(s,a) + \gamma \sum_{s'} T(s'|s,a) V_t(s') \right) \leq 3\epsilon_1. \tag{6}$$

Observe that K_t is defined by the true transition and reward functions T and R, and therefore cannot be known to the learner.

Now, consider the following statement:

Assumption A1 *Suppose an attempted update of state-action pair (s,a) occurs at time t, and that the m most recent experiences of (s,a) happened at times $k_1 < k_2 < \cdots < k_m = t$. If $(s,a) \notin K_{k_1}$ then the attempted update will be successful.*

During any given infinite-length execution of Delayed Q-learning, the statement ($A1$) may be true (all attempted updates with $(s,a) \notin K_{k_1}$ are successful) or it may be broken (some unsuccessful update may occur when $(s,a) \notin K_{k_1}$). When $(s,a) \notin K_{k_1}$, as above, our value function estimate $Q(s,a)$ is very inconsistent with our other value function estimates. Thus, we would expect our next attempted update to succeed. The next lemma shows this intuition is valid. Specifically, with probability at least $1-\delta/3$, $A1$ will be true. We are now ready to specify a value for m:

$$m := \frac{\ln\left(3SA(1+SA\kappa)/\delta\right)}{2{\epsilon_1}^2(1-\gamma)^2}. \tag{7}$$

Lemma 1 *The probability that $A1$ is violated during execution of Delayed Q-learning is at most $\delta/3$.*

Proof sketch: Fix any timestep k_1 (and the complete history of the agent up to k_1) satisfying: $(s,a) \notin K_{k_1}$ is to be experienced by the agent on timestep k_1 and if (s,a) is experienced $m-1$ more times after timestep k_1, then an attempted update will result. Let $\mathcal{Q} = [(s[1], r[1]), \ldots, (s[m], r[m])] \in (S \times \mathbb{R})^m$ be any sequence of m next-state and immediate reward tuples. Due to the Markov assumption, whenever the agent is in state s and chooses action a, the resulting next-state and immediate reward are chosen independently of the history of the agent. Thus, the probability that (s,a) is experienced $m-1$ more times and that the resulting next-state and immediate reward sequence equals $\mathcal{Q}$ is at most the probability that $\mathcal{Q}$ is obtained by m independent draws from the transition and reward distributions (for (s,a)). Therefore, it suffices to prove this lemma by showing that the probability that a random sequence $\mathcal{Q}$ could cause an unsuccessful update of (s,a) is at most $\delta/3$. We prove this statement next.

Suppose that m rewards, $r[1], \ldots, r[m]$, and m next states, $s[1], \ldots, s[m]$, are drawn independently from the reward and transition distributions, respectively, for (s,a). By a straightforward application of the Hoeffding bound (with random variables $X_i := r[i] + \gamma V_{k_1}(s[i])$), it can be shown that our choice of m guarantees that $(1/m)\sum_{i=1}^m (r[i] + \gamma V_{k_1}(s[i])) - E[X_1] < \epsilon_1$ holds with probability at least $1 - \delta/(3SA(1+SA\kappa))$. If it does hold and an attempted update is performed for (s,a) using these m samples, then the resulting update will succeed. To see the claim's validity, suppose that (s,a) is experienced at times $k_1 < k_2 < \cdots < k_m = t$ and at time k_i the agent is transitioned to state $s[i]$ and receives reward $r[i]$ (causing an attempted update at time t). Then, we have that

$$\begin{aligned} Q_t(s,a) - \left(\frac{1}{m} \sum_{i=1}^{m} (r[i] + \gamma V_{k_i}(s[i])) \right) \\ > \quad Q_t(s,a) - E[X_1] - \epsilon_1 \ > \ 2\epsilon_1. \end{aligned}$$

We have used the fact that $V_{k_i}(s') \leq V_{k_1}(s')$ for all s' and $i = 1, \ldots, m$. Therefore, with high probability, Condition 2 will be satisfied and the attempted update of $Q(s,a)$ at time k_m will succeed.

Finally, we extend our argument, using the union bound, to all possible timesteps k_1 satisfying the condition above. The number of such timesteps is bounded by the same bound we showed for the number of attempted updates ($SA(1+SA\kappa)$). $\square$

The next lemma states that, with high probability, Delayed Q-learning will maintain optimistic Q-values.

Lemma 2 *During execution of Delayed Q-learning, $Q_t(s,a) \geq Q^*(s,a)$ holds for all timesteps t and state-action pairs (s,a), with probability at least $1-\delta/3$.*

Proof sketch: It can be shown, by a similar argument as in the proof of Lemma 1, that $(1/m)\sum_{i=1}^m (r_{k_i} + \gamma V^*(s_{k_i})) > Q^*(s,a) - \epsilon_1$ holds, for all attempted updates, with probability at least $1-\delta/3$. Assuming this equation does hold, the proof is by induction on the timestep t. For the base case, note that $Q_1(s,a) = 1/(1-\gamma) \geq Q^*(s,a)$ for all (s,a). Now, suppose the claim holds for all timesteps less than or equal to t. Thus, we have that $Q_t(s,a) \geq Q^*(s,a)$, and $V_t(s) \geq V^*(s)$ for all

(s,a). Suppose s is the tth state reached and a is the action taken at time t. If it doesn't result in an attempted update or it results in an unsuccessful update, then no Q-value estimates change, and we are done. Otherwise, by Equation 1, we have that $Q_{t+1}(s,a) = (1/m)\sum_{i=1}^{m}(r_{k_i}+\gamma V_{k_i}(s_{k_i})) + \epsilon_1 \geq (1/m)\sum_{i=1}^{m}(r_{k_i}+\gamma V^*(s_{k_i})) + \epsilon_1 \geq Q^*(s,a)$, by the induction hypothesis and an application of the equation from above. □

Lemma 3 *If Assumption A1 holds, then the following statement holds: If an unsuccessful update occurs at time t and* $\texttt{LEARN}_{t+1}(s,a) = \texttt{false}$*, then* $(s,a) \in K_{t+1}$.

Proof: Suppose an attempted update of (s,a) occurs at time t. Let $s_{k_1}, s_{k_2}, \ldots, s_{k_m}$ be the m most recent next-states resulting from executing action a from state s at times $k_1 < k_2 < \cdots < k_m = t$, respectively. By $A1$, if $(s,a) \notin K_{k_1}$, then the update will be successful. Now, suppose that $(s,a) \in K_{k_1}$ but that $(s,a) \notin K_{k_i}$ for some $i \in \{2,\ldots,m\}$. In this case, the attempted update at time k_m may be unsuccessful. However, some Q-value estimate was successfully updated between time k_1 and time k_m (otherwise K_{k_1} would equal K_{k_1}). Thus, by the rules of Section 4.3, $\texttt{LEARN}(s,a)$ will be set to $\texttt{true}$ after this unsuccessful update ($\texttt{LEARN}_{t+1}(s,a)$ will be true). □

The following lemma bounds the number of timesteps t in which a state-action pair $(s,a) \notin K_t$ is experienced.

Lemma 4 *The number of timesteps t such that a state-action pair $(s,a) \notin K_t$ is experienced is at most $2mSA\kappa$.*

Proof: Suppose $(s,a) \notin K_t$ is experienced at time t and $\texttt{LEARN}_t(s,a) = \texttt{false}$ (implying the last attempted update was unsuccessful). By Lemma 3, we have that $(s,a) \in K_{t'+1}$ where t' was the time of the last attempted update of (s,a). Thus, some successful update has occurred since time $t'+1$. By the rules of Section 4.3, we have that $\texttt{LEARN}(s,a)$ will be set to $\texttt{true}$ and by $A1$, the next attempted update will succeed.

Now, suppose that $(s,a) \notin K_t$ is experienced at time t and $\texttt{LEARN}_t(s,a) = \texttt{true}$. Within at most m more experiences of (s,a), an attempted update of (s,a) will occur. Suppose this attempted update takes place at time q and that the m most recent experiences of (s,a) happened at times $k_1 < k_2 < \cdots < k_m = q$. By $A1$, if $(s,a) \notin K_{k_1}$, the update will be successful. Otherwise, if $(s,a) \in K_{k_1}$, then some successful update must have occurred between times k_1 and t (since $K_{k_1} \neq K_t$). Hence, even if the update is unsuccessful, $\texttt{LEARN}(s,a)$ will remain $\texttt{true}$, $(s,a) \notin K_{q+1}$ will hold, and the next attempted update of (s,a) will be successful.

In either case, if $(s,a) \notin K_t$, then within at most $2m$ more experiences of (s,a), a successful update of $Q(s,a)$ will occur. Thus, reaching a state-action pair not in K_t at time t will happen at most $2mSA\kappa$ times. □

We will make use of the following lemma from Strehl and Littman (2005).

Lemma 5 (Generalized Induced Inequality) *Let M be an MDP, K a set of state-action pairs, M' an MDP equal to M on K (identical transition and reward functions), π a policy, and T some positive integer. Let A_M be the event that a state-action pair not in K is encountered in a trial generated by starting from state s and following π for T timesteps in M. Then,*

$$V^{\pi}_{M}(s,T) \geq V^{\pi}_{M'}(s,T) - \Pr(A_M)/(1-\gamma).$$

5.2.1. Proof of the Main Result

Proof of Theorem 1: Suppose Delayed Q-learning is run on MDP M. We assume that $A1$ holds and that $Q_t(s,a) \geq Q^*(s,a)$ holds for all timesteps t and state-action pairs (s,a). The probability that either one of these assumptions is broken is at most $2\delta/3$, by Lemmas 1 and 2.

Consider timestep t during learning. Let $\mathcal{A}_t$ be the non-stationary policy being executed by the learning algorithm. Let π_t be the current greedy policy, that is, for all states s, $\pi_t(s) = \operatorname{argmax}_a Q_t(s,a)$. Let s_t be the current state, occupied by the agent at time t. We define a new MDP, M'. This MDP is equal to M on K_t (identical transition and reward functions). For $(s,a) \notin K_t$, we add a distinguished state $S_{s,a}$ to the state space of M' and a transition of probability one to that state from s when taking action a. Furthermore, $S_{s,a}$ self-loops on all actions with reward $[Q_t(s,a)-R(s,a)](1-\gamma)/\gamma$ (so that $V^{\pi}_{M'}(S_{s,a}) = [Q_t(s,a)-R(s,a)]/\gamma$ and $Q^{\pi}_{M'}(s,a) = Q_t(s,a)$, for any policy π). Let $T = O(\frac{1}{1-\gamma}\ln\frac{1}{\epsilon_2(1-\gamma)})$ be large enough so that $|V^{\pi_t}_{M'}(s_t,T) - V^{\pi_t}_{M'}(s_t)| \leq \epsilon_2$ (see Lemma 2 of Kearns and Singh (2002)). Let $\Pr(A_M)$ denote the probability of reaching a state-action pair (s,a) not in K_t, while executing policy $\mathcal{A}_t$ from state s_t in M for T timesteps. Let $\Pr(U)$ denote the probability of the algorithm performing a successful update on some state-action pair (s,a), while executing policy $\mathcal{A}_t$ from state s_t in M for T timesteps. We have that

$$\begin{aligned} & V_M^{\mathcal{A}_t}(s_t, T) \\ \geq \quad & V_{M'}^{\mathcal{A}_t}(s_t, T) - \Pr(A_M)/(1-\gamma) \\ \geq \quad & V_{M'}^{\pi_t}(s_t, T) - \Pr(A_M)/(1-\gamma) - \Pr(U)/(1-\gamma) \\ \geq \quad & V_{M'}^{\pi_t}(s_t) - \epsilon_2 - (\Pr(A_M) + \Pr(U))/(1-\gamma). \end{aligned}$$

The first step above follows from Lemma 5[3]. The second step follows from the fact that $\mathcal{A}_t$ behaves identically to π_t as long as no Q-value estimate updates are performed. The third step follows from the definition of T above.

Now, consider two mutually exclusive cases. First, suppose that $\Pr(A_M) + \Pr(U) \geq \epsilon_2(1-\gamma)$, meaning that an agent following $\mathcal{A}_t$ will either perform a successful update in T timesteps, or encounter some $(s,a) \notin K_t$ in T timesteps with probability at least $\epsilon_2(1-\gamma)/2$ (since $\Pr(A_M \text{ or } U) \geq (\Pr(A_M) + \Pr(U))/2$). The former event cannot happen more than $SA\kappa$ times. By assumption, the latter event will happen no more than $2mSA\kappa$ times (see Lemma 4). Define $\zeta = (2m+1)SA\kappa$. Using the Hoeffding bound, after $O(\frac{\zeta T}{\epsilon_2(1-\gamma)} \ln 1/\delta)$ timesteps where $\Pr(A_M) + \Pr(U) \geq \epsilon_2(1-\gamma)$, every state-action pair will have been updated $1/(\epsilon_1(1-\gamma))$ times, with probability at least $1-\delta/3$, and no futher updates will be possible. This fact implies that the number of timesteps t such that $\Pr(A_M) + \Pr(U) \geq \epsilon_2(1-\gamma)$ is bounded by $O(\frac{\zeta T}{\epsilon_2(1-\gamma)} \ln 1/\delta)$, with high probability.

Next, suppose that $\Pr(A_M) + \Pr(U) < \epsilon_2(1-\gamma)$. We claim that the following holds for all states s:

$$0 < V_t(s) - V_{M'}^{\pi_t}(s) \leq \frac{3\epsilon_1}{1-\gamma}. \qquad (8)$$

Recall that for all (s,a), either $Q_t(s,a) = Q_{M'}^{\pi_t}(s,a)$ (when $(s,a) \notin K_t$), or $Q_t(s,a) - (R(s,a) + \gamma \sum_{s'} T(s'|s,a) V_t(s')) \leq 3\epsilon_1$ (when $(s,a) \in K_t$). Note that $V_{M'}^{\pi_t}$ is the solution to the following set of equations:

$$\begin{aligned} V_{M'}^{\pi_t}(s) &= R(s,\pi_t(s)) + \gamma \sum_{s' \in S} T(s'|s,\pi_t(s)) V_{M'}^{\pi_t}(s'), \\ & \qquad \qquad \text{if } (s,\pi_t(s)) \in K, \\ V_{M'}^{\pi_t}(s) &= Q_t(s,\pi_t(s)), \qquad \text{if } (s,\pi_t(s)) \notin K. \end{aligned}$$

The vector V_t is the solution to a similar set of equations except with some additional positive reward terms, each bounded by $3\epsilon_1$. Using these facts, we have that

$$\begin{aligned} & V_M^{\mathcal{A}_t}(s_t) \geq V_M^{\mathcal{A}_t}(s_t, T) \\ \geq \quad & V_{M'}^{\pi_t}(s_t, T) - \epsilon_2 - (\Pr(A_M) + \Pr(U))/(1-\gamma) \\ \geq \quad & V_{M'}^{\pi_t}(s_t) - \epsilon_2 - \epsilon_2 \\ \geq \quad & V_t(s_t) - 3\epsilon_1/(1-\gamma) - 2\epsilon_2 \\ \geq \quad & V^*(s_t) - 3\epsilon_1/(1-\gamma) - 2\epsilon_2. \end{aligned}$$

The third step follows from the fact that $\Pr(A_M) + \Pr(U) < \epsilon_2(1-\gamma)$ and the fourth step from Equation 8. The last step made use of our assumption that $V_t(s_t) \geq V^*(s_t)$ always holds.

Finally, by setting $\epsilon_1 := \epsilon(1-\gamma)/9$ and $\epsilon_2 := \epsilon/3$, we have that

$$V_{\mathcal{A}_t}^{\pi_t}(s_t, T) \geq V^*(s_t) - \epsilon$$

is true for all but $O(\frac{\zeta T}{\epsilon_2(1-\gamma)} \ln 1/\delta)$

$$= O\left(\frac{SA}{\epsilon^4 (1-\gamma)^8} \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)} \ln \frac{SA}{\delta \epsilon (1-\gamma)} \right)$$

timesteps, with probability at least $1-\delta$. We guarantee a failure probability of at most δ by bounding the three sources of failure: from Lemmas 1, 2, and from the above application of Hoeffding's bound. Each of these will fail with probability at most $\delta/3$. □

Ignoring log factors, the best sample complexity bound previously proven has been

$$\tilde{O}\left(\frac{S^2 A}{\epsilon^3 (1-\gamma)^3} \right)$$

for the $R_{\max}$ algorithm as analyzed by Kakade (2003). Using the notation of Kakade (2003)[4], our bound of Theorem 1 reduces to

$$\tilde{O}\left(\frac{SA}{\epsilon^4 (1-\gamma)^4} \right).$$

It is clear that there is no strict improvement of the bounds, since a factor of S is being traded for one of $1/(\epsilon(1-\gamma))$. Nonetheless, to the extent that the dependence on S and A is of primary importance, this tradeoff is a net improvement. We also note that the best lower bound known for the problem, due to Kakade (2003), is $\tilde{\Omega}\left(SA/(\epsilon(1-\gamma))\right)$.

Our analysis of Delayed Q-learning required that γ be less than 1. The analyses of Kakade (2003) and Kearns and Singh (2002), among others, also considered the

[3]Lemma 5 is valid for all policies, including non-stationary ones.

[4]The use of *normalized* value functions reduces the dependence on $1/(1-\gamma)$.

case of $\gamma = 1$. Here, instead of evaluating a policy with respect to the infinite horizon, only the next H action-choices of the agent contribute to the value function. See Kakade (2003) for a discussion of how to evaluate hard horizon policies in an online exploration setting. For completeness, we also analyzed a version of Delayed Q-learning that works in this setting. We find that the agent will follow an ϵ-optimal policy for horizon H on all but $\tilde{O}(SAH^5/\epsilon^4)$ timesteps, with probability at least $1-\delta$. In terms of the dependence on the number of states (S), this bound is an improvement (from quadratic to linear) over previous bounds.

6. Related Work

There has been a great deal of theoretical work analyzing RL algorithms. Early results include proving that under certain conditions various algorithms can, in the limit, compute the optimal value function from which the optimal policy can be extracted (Watkins & Dayan, 1992). These convergence results make no performance guarantee after only a finite amount of experience. Even-Dar and Mansour (2003) studied the convergence rate of Q-learning. They showed that, under a certain assumption, Q-learning converges to a near-optimal value function in a polynomial number of timesteps. The result requires input of an exploration policy that, with high probability, tries every state-action pair every L timesteps (for some polynomial L). Such a policy may be hard to find in some MDPs and is impossible in others. The work by Fiechter (1994) proves that efficient learning (PAC) is achievable, via a model-based algorithm, when the agent has an action that *resets* it to a distinguished start state.

Other recent work has shown that various model-based algorithms, including E^3 (Kearns & Singh, 2002), $R_{\max}$ (Brafman & Tennenholtz, 2002), and MBIE (Strehl & Littman, 2005), are PAC-MDP. The bound from Theorem 1 improves upon those bounds when only the dependence of S and A is considered. Delayed Q-learning is also significantly more computationally efficient than these algorithms.

Delayed Q-learning can be viewed as an approximation of the real-time dynamic programming algorithm (Barto et al., 1995), with an added exploration bonus (of ϵ_1). The algorithm and its analysis are also similar to phased Q-learning and its analysis (Kearns & Singh, 1999). In both of the above works, exploration is not completely dealt with. In the former, the transition matrix is given as input to the agent. In the latter, an idealized exploration policy, one that samples every state-action pair simultaneously, is assumed to be provided to the agent.

7. Conclusion

We presented Delayed Q-learning, a provably efficient model-free reinforcement-learning algorithm. Its analysis solves an important open problem in the community. Future work includes closing the gap between the upper and lower bounds on PAC-MDP learning (see Section 5.2.1). More important is how to extend the results, using generalization, to richer world models with an infinite number of states and actions.

Acknowledgments

Thanks to the National Science Foundation (IIS-0325281). We also thank Yishay Mansour, Sham M. Kakade, Satinder Singh, and our anonymous reviewers for suggestions. Eric Wiewiora contributed to this research as an intern at TTI-Chicago.

References

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*, 81–138.

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

Even-Dar, E., & Mansour, Y. (2003). Learning rates for Q-learning. *Journal of Machine Learning Research*, *5*, 1–25.

Fiechter, C.-N. (1994). Efficient reinforcement learning. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory* (pp. 88–97). Association of Computing Machinery.

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning*. Doctoral dissertation, Gatsby Computational Neuroscience Unit, University College London.

Kearns, M., & Singh, S. (1999). Finite-sample convergence rates for Q-learning and indirect algorithms. *Advances in Neural Information Processing Systems 11* (pp. 996–1002). The MIT Press.

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

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

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. The MIT Press.

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