Journal of Machine Learning Research 7 (2006) 1079–1105 Submitted 2/05; Published 6/06

Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems*

Eyal Even-Dar EVENDAR@SEAS.UPENN.EDU
Department of Information and Computer Science
University of Pennsylvania
Philadelphia, PA 19104

Shie Mannor SHIE@ECE.MCGILL.CA
Department of Electrical & Computer Engineering
McGill University
H3A-2A7 Québec, Canada

Yishay Mansour MANSOUR@CS.TAU.AC.IL
School of Computer Science
Tel-Aviv University
Tel-Aviv, 69978, Israel

Editor: Sridhar Mahadevan

Abstract

We incorporate statistical confidence intervals in both the multi-armed bandit and the reinforcement learning problems. In the bandit problem we show that given n arms, it suffices to pull the arms a total of $O\left((n/\varepsilon^2)\log(1/\delta)\right)$ times to find an ε-optimal arm with probability of at least $1-\delta$. This bound matches the lower bound of Mannor and Tsitsiklis (2004) up to constants. We also devise action elimination procedures in reinforcement learning algorithms. We describe a framework that is based on learning the confidence interval around the value function or the Q-function and eliminating actions that are not optimal (with high probability). We provide a model-based and a model-free variants of the elimination method. We further derive stopping conditions guaranteeing that the learned policy is approximately optimal with high probability. Simulations demonstrate a considerable speedup and added robustness over ε-greedy Q-learning.

1. Introduction

Two of the most studied problems in control, decision theory, and learning in unknown environment are the multi-armed bandit (MAB) and reinforcement learning (RL). In this paper we consider both models under the probably approximately correct (PAC) settings and study several important questions arising in this model. The first question is when can an agent stop learning and start exploiting using the knowledge it obtained. The second question is which strategy leads to minimal learning time. Since the multi-armed bandit setup is simpler, we start by introducing it and later describe the reinforcement learning problem.

The Multi-armed bandit problem is one of the classical problems in decision theory and control. There is a number of alternative arms, each with a stochastic reward whose probability distribution is initially unknown. We try these arms in some order, which may depend on the sequence of rewards

*. Preliminary and partial results from this work appeared as extended abstracts in COLT 2002 and ICML 2003.

that have been observed so far. A common objective in this context is to find a policy for choosing the next arm to be tried, under which the sum of the expected rewards comes as close as possible to the ideal reward, i.e., the expected reward that would be obtained if we were to try the "best" arm at all times. One of the attractive features of the multi-armed bandit problem is that despite its simplicity, it encompasses many important decision theoretic issues, such as the tradeoff between exploration and exploitation.

The multi-armed bandit problem has been widely studied in a variety of setups. The problem was first considered in the 50's in the seminal work of Robbins (1952) that derives strategies that asymptotically attain an average reward that converges in the limit to the reward of the best arm. The multi-armed bandit problem was later studied in discounted, Bayesian, Markovian, expected reward, and adversarial setups. (See Berry and Fristedt, 1985, for a review of the classical results on the multi-armed bandit problem.) Most of the research so far has considered the expected regret, and devised strategies for minimizing it. The seminal work of Lai and Robbins (1985) provides tight bounds as a function of the Kullback-Leibler divergence between the arms reward distribution, and a logarithmic growth with the number of steps. The bounds of Lai and Robbins (1985) were shown to be efficient, in the sense that the convergence rates are optimal. The adversarial multi-armed bandit problem was considered in Auer et al. (1995, 2002), where it was shown that the expected regret grows proportionally to the square root of the number of steps.

We consider the classical multi-armed bandit problem, but rather than looking at the expected regret, we develop PAC style bounds. The agent's goal is to find, with high probability, a near optimal arm, namely, with probability at least $1-\delta$ output an ε-optimal arm. This naturally abstracts the case where the agent needs to choose one specific arm, and it is given only limited exploration initially. Our main complexity criterion, in addition to correctness, is the number of steps taken by the algorithm, which can be viewed as pure exploration steps. This is in contrast to most of the results for the multi-armed bandit problem, where the main aim is to maximize the expected cumulative reward while both exploring and exploiting. Therefore, methods which balance between exploration and exploitation such as softmax, and ε-greedy are not comparable to our methods. Following our initial conference publication, a lower bound on the number of steps needed to obtain a PAC solution was developed in Mannor and Tsitsiklis (2004); it matches the upper bound we develop in this paper.

The MAB problem models a situation where the environment is static and the same decision has to be made repeatedly. In many cases of practical interest, the model should represent a situation where the state of the system changes with time. This is encompassed in the Markov decision process model (MDP), that has been the subject of intensive research since the 1950's. When the model is known, and learning is not required, there are several standard methods for calculating the optimal policy - linear programming, value iteration, policy iteration, etc.; see Puterman (1994) for a review. When the model is not known a-priori, a *learning* scheme is needed. RL has emerged in the recent decade as unified discipline for adaptive control of dynamic environments (e.g., Sutton and Barto, 1998, Bertsekas and Tsitsiklis, 1996). A common problem with many RL algorithms is a slow convergence rate, even for relatively small problems. For example, consider the popular Q-learning algorithm (Watkins, 1989) which is essentially an asynchronous stochastic approximation algorithm (Bertsekas and Tsitsiklis, 1996). Generic convergence rate bounds for stochastic approximation (e.g., Borkar and Meyn, 2000) or specific rates for Q-learning (see, Kearns and Singh, 1998, Even-Dar and Mansour, 2003) are somewhat disappointing. However, the generic convergence rate is shown there to be almost tight for several particularly bad scenarios. The question that we ask

is: When is enough information gathered? When can the learning agent declare with reasonable confidence that the policy discovered is optimal, or at least approximately optimal? To summarize the differences, we are not concerned in the generic convergence rate (which must be slow), but we are rather interested in supplying rates which will be adjusted to the specific MDP parameters and as a result are much better for certain problems.

The problem of obtaining stopping conditions for learning in MDPs is a fundamental problem in RL. As opposed to supervised learning problems where typically a data set is given to the learner who has to commit to a classifier (or regressor in regression problem), in RL the decision maker can continue its interaction with the environment and obtain additional samples. The stopping rules that are currently employed in practice are based on ad-hoc rules, and may lead to premature stopping or to overly long trajectories.

When an action in a certain state can be determined to *not* belong to the optimal policy in an MDP, it can be discarded and disregarded in both planning and learning. This idea, commonly known as action elimination (AE), was proposed by MacQueen (1966) in the context of planning when the MDP parameters are known. In the planning case AE serves two purposes: reduce the size of the action sets to be searched at every iteration; identify optimal policies when there is a unique optimal policy. (In value iteration this is the only way to reach optimal policy rather than ε-optimal policy.) AE procedures are standard practice in solving large practical MDPs and are considered state-of-the-art; see Puterman (1994) for more details. We consider AE in the *learning* context when the model is not known a-priori.

In many applications the computational power is available but sampling of the environment is expensive. By eliminating sub-optimal actions early in the learning process, the total amount of sampling is reduced, leading to spending less time on estimating the parameters of sub-optimal actions. The main motivation for applying AE in RL is reducing the amount of samples needed from the environment. In addition to that, AE in RL enjoys the same advantages as in MDPs - convergence rate speedup and possibility to find an optimal policy (rather than ε-optimal).

Overview of the paper

After defining the settings in Section 2, we consider the MAB problem in Section 3. We start from a naive algorithm for the MAB problem, and present two improved algorithms. The first algorithm in the bandit settings, *Successive Elimination*, has the potential to exhibit an improved behavior in cases where the differences between the expected rewards of the optimal arm and sub-optimal arms are much larger than ε. The second algorithm, *Median Elimination*, achieves a better dependence on the number of arms. Namely, the total number of arm trials is $O(n/\varepsilon^2 \log(1/\delta))$, which improves the naive bound by a factor of $\log n$, and matches the lower bounds given in Mannor and Tsitsiklis (2004).

In Section 4 we consider AE in RL. The underlying idea is to maintain upper and lower estimates of the value (or Q) function. When the expected upper estimate of the return of a certain action falls below the expected lower estimate of another action, the obviously inferior action is eliminated. We suggest both, a model-based and a Q-learning style AE algorithms. The upper and lower bounds are based on a large deviations inequality, so that when an action is eliminated, it is not optimal with high probability.

Stopping conditions that are based on generic convergence rate bounds (as in Even-Dar and Mansour, 2003) are overly conservative. We suggest a stopping time based on the difference be-

tween the upper and lower bounds of the value (or Q) function. We show that if the difference is small, then the greedy policy with respect to the lower estimate is almost optimal.

In Section 5 we present the results of several experiments with AE in toy problems as well as in non-trivial problems. Significant speedup with negligible computational overhead is observed as compared to ε-greedy Q-learning.

2. Model and Preliminaries

In this section we define the models considered in this paper. We start from the MAB model in Section 2.1. We then describe the MDP model in Section 2.2. While both models are well studied we prefer to recapitulate them in the PAC setup, to avoid confusion. We finally recall Hoeffding's inequality which is a central tool in this work in Section 2.3.

2.1 Multi-Armed Bandit

The model is comprised of a set of arms A with $n = |A|$. When sampling arm $a \in A$ a reward which is a random variable $R(a)$ is received. We assume that the reward is binary, i.e., for every arm $a \in A$ the reward $R(a) \in \{0,1\}$ (all the results apply without change if the reward is bounded in $[0,1]$ and in general as long as the reward is bounded with appropriate modifications). Denote the arms by $a_1, \cdots, a_n$ and $p_i = \mathbb{E}[R(a_i)]$. For simplicity of notations we enumerate the arms according to their expected reward $p_1 > p_2 > ... > p_n$.

An arm with the highest expected reward is called the *best arm*, and denoted by a^*, and its expected reward r^* is the *optimal reward*. An arm whose expected reward is strictly less than r^*, the expected reward of the best arm, is called a *non-best arm*. An arm a is called an ε-*optimal arm* if its expected reward is at most ε from the optimal reward, i.e., $\mathbb{E}[R(a)] \geq r^* - \varepsilon$.

An algorithm for the MAB problem, at each time step t, samples an arm a_t and receives a reward r_t (distributed according to $R(a_t)$). When making its selection the algorithm may depend on the history (i.e., the actions and rewards) up to time $t-1$. Eventually the algorithm must commit to a single arm and select it.

Next we define the desired properties of an algorithm formally.

Definition 1 *An algorithm is a* (ε, δ)*-PAC algorithm for the multi armed bandit with* sample complexity T*, if it outputs an* ε*-optimal arm,* a'*, with probability at least* $1-\delta$*, when it terminates, and the number of time steps the algorithm performs until it terminates is bounded by* T.

Remark 2 The MAB algorithm may terminate before T steps passed. The sample complexity we consider is the complexity of the *worst* trajectory. The expected sample complexity (where the expectation is taken with respect to both the model and the algorithm) was considered in Mannor and Tsitsiklis (2004). The expected sample complexity behaves like $\Omega((n+\log(1/\delta))/\varepsilon^2)$, which is different than the complexity we prove below in Theorem 10. We note that the running time of the algorithm from Mannor and Tsitsiklis (2004) is not bounded in the worst case.

2.2 Markov Decision Processes

We define an MDP as follows:

Definition 3 *A Markov Decision process (MDP) M is a 4-tuple (S,A,P,R), where S is a set of the states, A is a set of actions, $P^a_{s,s'}$ is the transition probability from state s to state s' when performing action $a \in A$ in state s, and $R(s,a)$ is the reward received when performing action a in state s.*

A strategy for an MDP assigns, at each time t, for each state s a probability for performing action $a \in A$, given a history $F_{t-1} = \{s_1, a_1, r_1, ..., s_{t-1}, a_{t-1}, r_{t-1}\}$ which includes the states, actions and rewards observed until time $t-1$. While executing a strategy π we perform at time t action a_t in state s_t and observe a reward r_t (distributed according to $R(s_t,a_t)$), and the next state s_{t+1} distributed according to $P^{a_t}_{s_t,\cdot}$. We combine the sequence of rewards into a single value called the *return*. Our goal is to maximize the return. In this work we focus on the *discounted return*, which has a parameter $\gamma \in (0,1)$, and the discounted return of policy π is $V^\pi = \sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward observed at time t. We also consider the *finite horizon* return, $V^\pi = \sum_{t=0}^{H} r_t$ for a given horizon H.

We assume that $R(s,a)$ is non-negative and bounded by R_{max}, i.e., for every s,a : $0 \le R(s,a) \le R_{max}$. This implies that the discounted return is bounded by $V_{max} = R_{max}/(1-\gamma)$; for the finite horizon the return is bounded by HR_{max}. We define a value function for each state s, under policy π, as $V^\pi(s) = \mathbb{E}^\pi[\sum_{i=0}^{\infty} r_i \gamma^i]$, where the expectation is over a run of policy π starting at state s. We further denote the state-action value function as using action a in state s and then following π as:

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P^a_{s,s'} V^\pi(s').$$

Similarly, we define the value functions for the finite horizon model.

Let π^* be an optimal policy which maximizes the return from any start state. For discounted return criterion, there exists such a policy which is deterministic and stationary (see, e.g., Puterman, 1994). This implies that for any policy π and any state s we have $V^{\pi^*}(s) \ge V^\pi(s)$, and $\pi^*(s) = \text{argmax}_a(R(s,a) + \gamma(\sum_{s'} P^a_{s,s'} V^{\pi^*}(s'))$. We use V^* and Q^* for V^{π^*} and Q^{π^*}, respectively. We say that a policy π is ε-optimal if $\|V^* - V^\pi\|_\infty \le \varepsilon$. We also define the policy $Greedy(Q)$ as the policy that prescribes in each state the action that maximizes the Q-function in the state, i.e., $\pi(s) = \text{argmax}_a Q(s,a)$.

For a given trajectory let: $T^{s,a}$ be the set of times in which we perform action a in state s and $T^{s,a,s'}$ be a subset of $T^{s,a}$ in which we reached state s'. Also, $\#(s,a,t)$ is the number of times action a is performed in state s up to time t, i.e., $|T^{s,a} \cap \{1,2,3,\ldots,t\}|$. We similarly define $\#(s,a,s',t)$ as $|T^{s,a,s'} \cap \{1,2,3,\ldots,t\}|$. Next we define the empirical model at time t. Given that $\#(s,a,t) > 0$ we define the empirical next state distribution at time t as

$$\hat{P}^a_{s,s'} = \frac{\#(s,a,s',t)}{\#(s,a,t)} \quad \text{and} \quad \hat{R}(s,a) = \frac{\sum_{t \in T^{s,a}} r_t}{\#(s,a,t)}.$$

If $\#(s,a,t) = 0$ the empirical model and the reward can be chosen arbitrarily. We define the expectation of the empirical model as $\hat{\mathbb{E}}_{s,s',a}[V(s')] = \sum_{s' \in S} \hat{P}^a_{s,s'} V(s')$. To simplify the notations we omit s,a in the notations $\hat{\mathbb{E}}_{s'}$ whenever evident.

2.3 A Concentration Bound

We often use large deviation bounds in this paper. Since we assume boundedness we can rely on Hoeffding's inequality.

Lemma 4 *(Hoeffding, 1963) Let X be a set, D be a probability distribution on X, and $f_1,...,f_m$ be real-valued functions defined on X with $f_i : X \to [a_i,b_i]$ for $i = 1,...,m$, where a_i and b_i are real numbers satisfying $a_i < b_i$. Let $x_1,\ldots,x_m$ be independent identically distributed samples from D. Then we have the following inequality*

$$\begin{aligned}
\mathbf{P}\left[\frac{1}{m}\sum_{i=1}^{m} f_i(x_i) - \left(\frac{1}{m}\sum_{i=1}^{m}\int_{a_i}^{b_i} f_i(x)D(x)\right) \geq \varepsilon\right] &\leq e^{-\frac{2\varepsilon^2 m^2}{\sum_{i=1}^{m}(b_i-a_i)^2}} \\
\mathbf{P}\left[\frac{1}{m}\sum_{i=1}^{m} f_i(x_i) - \left(\frac{1}{m}\sum_{i=1}^{m}\int_{a_i}^{b_i} f_i(x)D(x)\right) \leq -\varepsilon\right] &\leq e^{-\frac{2\varepsilon^2 m^2}{\sum_{i=1}^{m}(b_i-a_i)^2}}.
\end{aligned}$$

Remark 5 We note that the boundedness assumption is not essential and can be relaxed in certain situations. We also note that sometimes tighter bounds can be obtained using the relative Chernoff bound (Angluin and Valiant, 1979).

3. PAC Bounds for the Multi-Armed Bandit Problem

In this section we investigate an (ε,δ)-PAC algorithms for the MAB problem. Such algorithms are required to output with probability $1-\delta$ an ε-optimal arm. We start with a naive solution that samples each arm $1/(\varepsilon/2)^2\ln(2n/\delta)$ and picks the arm with the highest empirical reward. The sample complexity of this naive algorithm is $O(n/\varepsilon^2\log(n/\delta))$. The naive algorithm is described in Algorithm 1. In Section 3.1 we consider an algorithm that eliminates one arm after the other. In Section 3.2 we finally describe the Median Elimination algorithm whose sample complexity is optimal in the worst case.

Input : $\varepsilon > 0$, $\delta > 0$
Output : An arm
foreach Arm $a \in A$ **do**
  Sample it $\ell = \frac{4}{\varepsilon^2}\ln(\frac{2n}{\delta})$ times;
  Let $\hat{p}_a$ be the average reward of arm a;
end
Output $a' = \arg\max_{a\in A}\{\hat{p}_a\}$;

Algorithm 1: Naive Algorithm

Theorem 6 *The algorithm* Naive(ε,δ) *is an* (ε,δ)*-PAC algorithm with arm sample complexity* $O\left((n/\varepsilon^2)\log(n/\delta)\right)$.

Proof The sample complexity is immediate from the definition of the algorithm. We now prove it is an (ε,δ)-PAC algorithm. Let a' be an arm for which $\mathbb{E}(R(a')) < r^* - \varepsilon$. We want to bound the probability of the event $\hat{p}_{a'} > \hat{p}_{a^*}$.

$$\begin{aligned}
P(\hat{p}_{a'} > \hat{p}_{a^*}) &\leq P\left(\hat{p}_{a'} > \mathbb{E}[R(a')] + \varepsilon/2 \text{ or } \hat{p}_{a^*} < r^* - \varepsilon/2\right) \\
&\leq P\left(\hat{p}_{a'} > \mathbb{E}[R(a')] + \varepsilon/2\right) + P\left(\hat{p}_{a^*} < r^* - \varepsilon/2\right) \\
&\leq 2\exp(-2(\varepsilon/2)^2\ell),
\end{aligned}$$

where the last inequality uses the Hoeffding inequality. Choosing $\ell = (2/\varepsilon^2)\ln(2n/\delta)$ assures that $P(\hat{p}_{a'} > \hat{p}_{a^*}) \leq \delta/n$. Summing over all possible a' we have that the failure probability is at most $(n-1)(\delta/n) < \delta$. ■

3.1 Successive Elimination

The successive elimination algorithm attempts to sample each arm a minimal number of times and eliminate the arms one after the other. To motivate the successive elimination algorithm, we first assume that the expected rewards of the arms are known, but the matching of the arms to the expected rewards is unknown. Let $\Delta_i = p_1 - p_i > 0$. Our aim is to sample arm a_i for $(1/\Delta_i^2)\ln(n/\delta)$ times, and then eliminate it. This is done in phases. Initially, we sample each arm $(1/\Delta_n^2)\ln(n/\delta)$ times. Then we eliminate the arm which has the lowest empirical reward (and never sample it again). At the i-th phase we sample each of the $n-i$ surviving arms

$$O\left(\left(\frac{1}{\Delta_{n-i}^2} - \frac{1}{\Delta_{n-i+1}^2}\right)\log\left(\frac{n}{\delta}\right)\right)$$

times and then eliminate the empirically worst arm. The algorithm described as Algorithm 2 below. In Theorem 7 we prove that the algorithm is $(0,\delta)$-PAC and compute its sample complexity.

Input : $\delta > 0$, bias of arms $p_1, p_2, \ldots, p_n$
Output : An arm
Set $S = A$; $t_i = (8/\Delta_i^2)\ln(2n/\delta)$; and $t_{n+1} = 0$, for every arm a: $\hat{p}_a = 0$, $i = 0$;
while $i < n-1$ **do**
  Sample every arm $a \in S$ for $t_{n-i} - t_{n-i+1}$ times;
  Let $\hat{p}_a$ be the average reward of arm a (in all rounds);
  Set $S = S \setminus \{a_{\min}\}$, where $a_{\min} = \arg\min_{a \in S}\{\hat{p}_a\}$, $i = i+1$;
end
Output S;

Algorithm 2: Successive Elimination with Known Biases

Theorem 7 *Suppose that $\Delta_i > 0$ for $i = 2,3,\ldots,n$. Then the Successive Elimination with Known Biases algorithm is an $(0,\delta)$-PAC algorithm and its arm sample complexity is*

$$O\left(\log\left(\frac{n}{\delta}\right)\sum_{i=2}^{n}\frac{1}{\Delta_i^2}\right). \tag{1}$$

Proof The sample complexity of the algorithm is as follows. In the first round we sample n arms t_n times. In the second round we sample $n-1$ arms $t_{n-1} - t_n$ times. In the kth round ($1 \leq k < n$) we sample $n-k+1$ arms for $t_{n-k} - t_{n-k+1}$ times. The total number of arms samples is therefore $t_2 + \sum_{i=2}^{n} t_i$ which is of the form (1).
We now prove that the algorithm is correct with probability at least $1-\delta$. Consider first a simplified

algorithm which is similar to the naive algorithm, suppose that each arm is pulled $8/(\Delta_2^2)\ln(2n/\delta)$ times. For every $2 \le i \le n-1$ we define the event

$$E_i = \left\{ \hat{p}_1^{t_j} \ge \hat{p}_i^{t_j} | \forall t_j \text{ s.t. } j \ge i \right\},$$

where $\hat{p}_i^{t_j}$ is the empirical value the ith arm at time t_j. If the events E_i hold for all $i > 1$ the algorithm is successful.

$$\begin{aligned}
\mathbf{P}[\text{not}(E_i)] &\le \sum_{j=i}^{n} \mathbf{P}[\hat{p_n}^{t_j} < \hat{p}_i^{t_j}] \\
&\le \sum_{j=i}^{n} 2\exp(-2(\Delta_i/2)^2 t_j) \le \sum_{j=i}^{n} 2\exp(-2(\Delta_i/2)^2 8/\Delta_j^2 \ln(2n/\delta)) \\
&\le \sum_{j=i}^{n} 2\exp(-\ln(4n^2/\delta^2)) \\
&\le (n-i+1)\delta^2/n^2 \le \frac{\delta}{n}.
\end{aligned}$$

Using the union bound over all E_i's we obtain that the simplified algorithm satisfies all E_i with probability at least $1-\delta$. Consider the original setup. If arm 1 is eliminated at time t_j for some is implies that some arm $i < j$ has higher empirical value at time t_j. The probability of failure of the algorithm is bounded by the probability of failure in the simplified setting. ■

Next, we relax the requirement that the expected rewards of the arms are known in advance, and introduce the Successive Elimination algorithm that works with any set of biases. The algorithm we present as Algorithm 3 finds the best arm (rather than ε-best) with high probability. We later explain in Remark 9 how to modify it to be an (ε, δ)-PAC algorithm.

Input : $\delta > 0$
Output : An arm
Set $t = 1$ and $S = A$;
Set for every arm a: $\hat{p}_a^1 = 0$;
Sample every arm $a \in S$ once and let $\hat{p}_a^t$ be the average reward of arm a by time t;
repeat
  Let $\hat{p}_{max}^t = \max_{a \in S} \hat{p}_a^t$ and $\alpha_t = \sqrt{\ln(cnt^2/\delta)/t}$, where c is a constant;
  foreach arm $a \in S$ such that $\hat{p}_{max}^t - \hat{p}_a^t \ge 2\alpha_t$ **do**
    set $S = S \setminus \{a\}$;
  end
  $t = t + 1$;
until $|S| > 1$;

Algorithm 3: Successive elimination with unknown biases

Theorem 8 *Suppose that $\Delta_i > 0$ for $i = 2, 3, \ldots, n$. Then the Successive Elimination algorithm (Algorithm 3) is a $(0, \delta)$-PAC algorithm, and with probability at least $1-\delta$ the number of samples*

is bounded by

$$O\left(\sum_{i=2}^{n}\frac{\ln(\frac{n}{\delta\Delta_i})}{\Delta_i^2}\right).$$

Proof Our main argument is that, at any time t and for any action a, the observed probability $\hat{p}_a^t$ is within α_t of the true probability p_a. For any time t and action $a \in S_t$ we have that,

$$\mathbf{P}[|\hat{p}_a^t - p_a| \geq \alpha_t] \leq 2e^{-2\alpha_t^2 t} \leq \frac{2\delta}{cnt^2}.$$

By taking the constant c to be greater than 4 and from the union bound we have that with probability at least $1-\delta/n$ for any time t and any action $a \in S_t$, $|\hat{p}_a^t - p_a| \leq \alpha_t$. Therefore, with probability $1-\delta$, the best arm is never eliminated. Furthermore, since α_t goes to zero as t increases, eventually every non-best arm is eliminated. This completes the proof that the algorithm is $(0,\delta)$-PAC.

It remains to compute the arm sample complexity. To eliminate a non-best arm a_i we need to reach a time t_i such that,

$$\hat{\Delta}_{t_i} = \hat{p}_{a_1}^{t_i} - \hat{p}_{a_i}^{t_i} \geq 2\alpha_{t_i}.$$

The definition of α_t combined with the assumption that $|\hat{p}_a^t - p_a| \leq \alpha_t$ yields that

$$\Delta_i - 2\alpha_t = (p_1 - \alpha_t) - (p_i + \alpha_t) \geq \hat{p}_1 - \hat{p}_i \geq 2\alpha_t,$$

which holds with probability at least $1-\frac{\delta}{n}$ for

$$t_i = O\left(\frac{\ln(n/\delta\Delta_i)}{\Delta_i^2}\right).$$

To conclude, with probability of at least $1-\delta$ the number of arm samples is $2t_2 + \sum_{i=3}^{n} t_i$, which completes the proof. ■

Remark 9 One can easily modify the successive elimination algorithm so that it is (ε,δ)-PAC. Instead of stopping when only one arm survives the elimination, it is possible to settle for stopping when either only one arm remains or when each of the k surviving arms were sampled $O(\frac{1}{\varepsilon^2}\log(\frac{k}{\delta}))$. In the latter case the algorithm returns the best arm so far. In this case it is not hard to show that the algorithm finds an ε-optimal arm with probability at least $1-\delta$ after

$$O\left(\sum_{i:\Delta_i>\varepsilon}\frac{\log(\frac{n}{\delta\Delta_i})}{\Delta_i^2} + \frac{N(\Delta,\varepsilon)}{\varepsilon^2}\log\left(\frac{N(\Delta,\varepsilon)}{\delta}\right)\right),$$

where $N(\Delta,\varepsilon) = |\{i \mid \Delta_i < \varepsilon\}|$ is the number of arms which are ε-optimal.

3.2 Median Elimination

The following algorithm substitutes the term $O(\log(1/\delta))$ for $O(\log(n/\delta))$ of the naive bound. The idea is to eliminate the worst half of the arms at each iteration. We do not expect the best arm to be empirically "the best", we only expect an ε-optimal arm to be above the median.

Input : $\varepsilon > 0, \delta > 0$
Output : An arm
Set $S_1 = A$, $\varepsilon_1 = \varepsilon/4$, $\delta_1 = \delta/2$, $\ell = 1$. **repeat**
  Sample every arm $a \in S_\ell$ for $1/(\varepsilon_\ell/2)^2 \log(3/\delta_\ell)$ times, and let $\hat{p}_a^\ell$ denote its empirical value;
  Find the median of $\hat{p}_a^\ell$, denoted by m_ℓ;
  $S_{\ell+1} = S_\ell \setminus \{a : \hat{p}_a^\ell < m_\ell\}$;
  $\varepsilon_{\ell+1} = \frac{3}{4}\varepsilon_\ell$; $\delta_{\ell+1} = \delta_\ell/2$; $\ell = \ell + 1$;
until $|S_\ell| = 1$;

Algorithm 4: Median Elimination

Theorem 10 *The Median Elimination(ε,δ) algorithm is an (ε, δ)-PAC algorithm and its sample complexity is*

$$O\left(\frac{n}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right).$$

First we show that in the ℓ-th phase the expected reward of the best arm in S_ℓ drops by at most ε_ℓ.

Lemma 11 *For the* Median Elimination(ε, δ) *algorithm we have that for every phase ℓ:*

$$\mathbf{P}[\max_{j \in S_\ell} p_j \leq \max_{i \in S_{\ell+1}} p_i + \varepsilon_\ell] \geq 1 - \delta_\ell.$$

Proof Without loss of generality we look at the first round and assume that p_1 is the reward of the best arm. We bound the failure probability by looking at the event $E_1 = \{\hat{p}_1 < p_1 - \varepsilon_1/2\}$, which is the case that the empirical estimate of the best arm is pessimistic. Since we sample sufficiently, we have that $\mathbf{P}[E_1] \leq \delta_1/3$.

In case E_1 does not hold, we calculate the probability that an arm j which is not an ε_1-optimal arm is empirically better than the best arm.

$$\mathbf{P}\left[\hat{p}_j \geq \hat{p}_1 \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2\right] \leq \mathbf{P}\left[\hat{p}_j \geq p_j + \varepsilon_1/2 \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2\right] \leq \delta_1/3$$

Let #bad be the number of arms which are not ε_1-optimal but are empirically better than the best arm. We have that $\mathbb{E}[\#\text{bad} \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2] \leq n\delta_1/3$. Next we apply Markov inequality to obtain,

$$\mathbf{P}[\#\text{bad} \geq n/2 \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2] \leq \frac{n\delta_1/3}{n/2} = 2\delta_1/3.$$

Using the union bound gives us that the probability of failure is bounded by δ_1. ■

Next we prove that arm sample complexity is bounded by $O((n/\varepsilon^2)\log(1/\delta))$.

Lemma 12 *The sample complexity of the* Median Elimination(ε, δ) *is* $O\left((n/\varepsilon^2)\log(1/\delta)\right)$.

Proof The number of arm samples in the ℓ-th round is $4n_\ell \log(3/\delta_\ell)/\varepsilon_\ell^2$. By definition we have that

1. $\delta_1 = \delta/2$; $\delta_\ell = \delta_{\ell-1}/2 = \delta/2^\ell$
2. $n_1 = n$; $n_\ell = n_{\ell-1}/2 = n/2^{\ell-1}$
3. $\varepsilon_1 = \varepsilon/4$; $\varepsilon_\ell = \frac{3}{4}\varepsilon_{\ell-1} = \left(\frac{3}{4}\right)^{\ell-1}\varepsilon/4$

Therefore we have

$$\begin{aligned}
\sum_{\ell=1}^{\log_2(n)} \frac{n_\ell \log(3/\delta_\ell)}{(\varepsilon_\ell/2)^2} &= 4\sum_{\ell=1}^{\log_2(n)} \frac{n/2^{\ell-1}\log(2^\ell 3/\delta)}{((\frac{3}{4})^{\ell-1}\varepsilon/4)^2} \\
&= 64\sum_{\ell=1}^{\log_2(n)} n(\frac{8}{9})^{\ell-1}(\frac{\log(1/\delta)}{\varepsilon^2} + \frac{\log(3)}{\varepsilon^2} + \frac{\ell\log(2)}{\varepsilon^2}) \\
&\leq 64\frac{n\log(1/\delta)}{\varepsilon^2}\sum_{\ell=1}^{\infty}(\frac{8}{9})^{\ell-1}(\ell C' + C) = O(\frac{n\log(1/\delta)}{\varepsilon^2})
\end{aligned}$$

■

Now we can prove Theorem 10.

Proof From Lemma 12 we have that the sample complexity is bounded by $O\left(n\log(1/\delta)/\varepsilon^2\right)$. By Lemma 11 we have that the algorithm fails with probability δ_i in each round so that over all rounds the probability of failure is bounded by $\sum_{i=1}^{\log_2(n)}\delta_i \leq \delta$. In each round we reduce the optimal reward of the surviving arms by at most ε_i so that the total error is bounded by $\sum_{i=1}^{\log_2(n)}\varepsilon_i \leq \varepsilon$. ■

4. Learning in MDPs

In this section we consider algorithms for the RL problem, which are based on the MAB algorithms presented above. We start from model-based learning in Section 4.1, where the parameters of the models are learned. We describe algorithms which are based on the successive elimination algorithm and provide stopping conditions for these algorithms. In Section 4.2 we consider model-free learning and suggest a version of the Q-learning algorithm that incorporates action elimination and stopping conditions. In Section 4.3 we analyze the batched sampling setting of Kearns and Singh (2002) and provide a mechanism that can use any (ε,δ)-MAB algorithm to enhance the performance of the Phased Q-learning introduced in Kearns and Singh (2002). We also provide a matching lower bound.

4.1 Model-Based Learning

In this section we focus on model-based learning. In model-based methods, we first learn the model, i.e., estimate the immediate reward and the next state distribution. Then by either value iteration, policy iteration, or linear programming on the learned (empirical) model, we find the exact optimal policy for the empirical model. If enough exploration is done, this policy is almost optimal for the true model. We note that there is an inherent difference between the finite horizon and the infinite discounted return. Technically, the finite horizon return is simpler than the discounted return, as one can apply the concentration inequality directly. We provide model-based algorithms for both cases.

4.1.1 FINITE HORIZON

Let us first recall the classical optimality equations for finite horizon:

$$\begin{aligned} V^H(s) &= \max_a \{R(s,a) + \mathbb{E}_{s'}[V^{H-1}(s')]\}, \quad H > 0 \\ V^0(s) &= \max_a R(s,a), \end{aligned}$$

where $V^H(s)$ is the optimal value function for horizon H. We often abuse notation by using $\mathbb{E}_{s'}$ instead of $\mathbb{E}_{s',a}$. Given the empirical model by time t we define the upper estimate $\overline{V}_\delta$, which will be shown to satisfy for every horizon k and every state s, $\overline{V}^k_\delta(s) \geq V^k(s)$ with high probability. For horizon H we define:

$$\overline{V}^H_\delta(s) = \max_a \left\{ \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[\overline{V}^{H-1}_\delta(s')] + HR_{max}\sqrt{\frac{\ln(\frac{c|S||A|H^2}{\delta})}{|T^{s,a}|}} \right\}, \quad H > 0 \tag{2}$$

$$\overline{V}^0_\delta(s) = \max_a \left\{ \hat{R}(s,a) + R_{max}\sqrt{\frac{\ln(\frac{c|S||A|}{\delta})}{|T^{s,a}|}} \right\}, \tag{3}$$

for some constant $c \geq 4$. Similarly to the upper bound $\overline{V}^H_\delta$, a lower bound may be defined where the R_{max} is replaces by $-R_{max}$. We call this estimate the lower estimate $\underline{V}^H_\delta$. The following Lemma proves that $\overline{V}^H_\delta$ is an upper estimation for any horizon and that $\underline{V}^H_\delta$ is a lower estimation.

Theorem 13 *We have that $\overline{V}^k_\delta(s) \geq V^k(s) \geq \underline{V}^k_\delta(s)$ for all states s and horizons k, with probability at least $1-\delta$.*

Proof We prove the claim by induction. For the base of the induction, by a simple use of Hoeffding inequality we have that for every state s $\overline{V}^0_\delta(s) \geq \max_a \hat{R}(s,a)$ with probability $1-\delta/(c|S||A|)$. Next we assume that the claim holds for $i \leq k$ and prove for $k+1$ and for every action a. By definition $\overline{V}^{k+1}_\delta(s)$ satisfies for every a that

$$\begin{aligned} \overline{V}^{k+1}_\delta(s) &\geq \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[\overline{V}^k_\delta(s')] + (k+1)R_{max}\sqrt{\frac{\ln(\frac{c|S||A|(k+1)^2}{\delta})}{|T^{s,a}|}} \\ &\geq \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[V^k(s')] + (k+1)R_{max}\sqrt{\frac{\ln(\frac{c|S||A|(k+1)^2}{\delta})}{|T^{s,a}|}}, \end{aligned}$$

where the second inequality follows from the inductive hypothesis. Note that V^k is not a random variable, so we can bound the last expression using Hoeffding's inequality. We arrive at:

$$\begin{aligned} \mathbf{P}\left\{ \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[V^k(s')] + (k+1)R_{max}\sqrt{\frac{\ln(\frac{c|S||A|(k+1)^2}{\delta})}{|T^{s,a}|}} < R(s,a) + \mathbb{E}_{s'}[V^k(s')] \right\} \\ \leq e^{\frac{-\ln(\frac{c|S||A|(k+1)^2}{\delta})|T^{s,a}|\left(\frac{(k+1)R_{max}}{\sqrt{|T^{s,a}|}}\right)^2}{((k+1)R_{max})^2}} = \frac{\delta}{c|S||A|(k+1)^2}. \end{aligned}$$

Therefore, we have that with high probability the following holds

$$\begin{aligned}\overline{V}_{\delta}^{k+1}(s) &\geq \max_a\{R(s,a)+\hat{\mathbb{E}}_{s'}[V^k(s')]+kR_{max}\sqrt{\frac{\ln(\frac{c|S||A|k^2}{\delta})}{|T^{s,a}|}}\}\\ &\geq \max_a\{R(s,a)+\mathbb{E}_{s'}[V^k(s')]\}\\ &= V^{k+1}(s).\end{aligned}$$

Using the union bound over all state-action pairs and all finite horizons k, we obtain that the failure probability is bounded by $\delta/2$ for $c \geq 4$. Repeating the same argument for the lower estimate and applying the union bound completes the proof. ■

Consequently, a natural early stopping condition is to stop sampling when $\|\overline{V}^H - \underline{V}^H\|_\infty < \varepsilon$. We do not provide an algorithm here, however a detailed algorithm will be given in the following subsection.

4.1.2 DISCOUNTED RETURN - INFINITE HORIZON

In this subsection, we provide upper and lower estimates of the value function V for the infinite horizon case. The optimal value is the solution of the set of the equations:

$$V^*(s) = \max_a\{R(s,a)+\gamma\mathbb{E}_{s'}[V^*(s')]\}, \quad s \in S.$$

As in Subsection 4.1.1, we provide an upper value function $\overline{V}_\delta$, which satisfies with high probability $\overline{V}_\delta(s) \geq V^*(s)$. We define $\overline{V}_\delta^t$ at time t as the solution of the set of equations:

$$\overline{V}_\delta^t(s) = \max_a\left\{\hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta^t(s')]+V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|)}}\right\}$$

for some positive constant c and $\overline{Q}_\delta^t$ as:

$$\overline{Q}_\delta^t(s,a) = \hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta(s')]+V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|}}.$$

Similarly, we define $\underline{V}_\delta^t$ and $\underline{Q}_\delta^t$ as:

$$\underline{V}_\delta^t(s) = \max_a\left\{\hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}\underline{V}_\delta(s')-V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|)}}\right\}$$

$$\underline{Q}_\delta^t(s,a) = \hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}[\underline{V}_\delta^t(s')]-V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|}}.$$

The next lemma shows that with high probability the upper and lower estimations are indeed correct.

Lemma 14 *With probability at least* $1-\delta$ *we have that* $\overline{Q}_\delta^t(s,a) \geq Q^*(s,a) \geq \underline{Q}_\delta^t(s,a)$ *for every state* s*, action* a *and time* t*.*

Proof Suppose we run a value iteration algorithm on the empirical model at time t. Let $\overline{V}_\delta^{t,k}$ be the kth iteration of the value function algorithm at time t, and let $\overline{Q}_\delta^{t,k}$ be the associated Q-function, that is

$$\overline{Q}_\delta^{t,k}(s,a) = \hat{R}(s,a) + \gamma \hat{\mathbb{E}}_{s'}[\overline{V}_\delta^{t,k}(s')] + V_{max}\sqrt{\frac{\ln(\frac{ct^2|S||A|}{\delta})}{|T^{s,a}|}}.$$

Assume that we start with $\overline{V}_\delta^{t,0} = V^*$. (The use of V^* is restricted to the proof and not used in the algorithm.) We need to prove that $\overline{Q}_\delta^t(s,a) \geq Q^*(s,a)$ for every s and a. Note that since the value iteration converges, $\overline{Q}_\delta^{t,k}$ converges to $\overline{Q}_\delta^t$. We prove by induction on the number of the iterations that by taking $\overline{V}_\delta^{t,0} = V^*$, with high probability for every k we have that $\overline{Q}_\delta^{t,k} \geq \overline{Q}_\delta^{t,k-1}$, i.e., $\mathbf{P}[\forall k\, \overline{Q}_\delta^k \geq \overline{Q}_\delta^{k-1}] \geq 1 - \frac{\delta}{ct^2}$. For the basis, since V^* is not a random variable we can apply Hoeffding's inequality and obtain that for every state action pair (s,a)

$$\begin{aligned}\mathbf{P}\Big\{\hat{R}(s,a) + \gamma\hat{\mathbb{E}}_{s'}[V^*(s')] + V_{max}\sqrt{\frac{\ln(\frac{ct^2|S||A|}{\delta})}{|T^{s,a}|}} < R(s,a) + \gamma \mathbb{E}_{s'}[V^*(s')]\Big\} \\ \leq e^{-\ln(\frac{ct^2|S||A|}{\delta})} = \frac{\delta}{ct^2|S||A|}.\end{aligned}$$

Since $\overline{V}_\delta^{t,0}(s) = V^*$ we have that $\overline{Q}_\delta^{t,1}(s,a) = \hat{R}(s,a) + \gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta^{t,0}(s')] + V_{max}\sqrt{\frac{\ln(\frac{ct^2|S||A|}{\delta})}{|T^{s,a}|}}$. Therefore, $\overline{Q}_\delta^{t,1} \geq \overline{Q}_\delta^{t,0}$ with probability $1 - \frac{\delta}{ct^2}$. For the induction step, we assume that the claim holds for $i < k$ and prove for k.

$$\overline{Q}_\delta^{t,k}(s,a) - \overline{Q}_\delta^{t,k-1}(s,a) \quad = \quad \gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta^{t,k-1}(s') - \overline{V}_\delta^{t,k-2}(s')].$$

Since $\overline{V}_\delta^{t,k-1}(s') = \max_a \overline{Q}_\delta^{t,k-1}(s',a)$ we have by the induction that for every s,

$$V_\delta^{t,k-1}(s) = \max_a \overline{Q}_\delta^{t,k-1}(s,a) \geq \max_a \overline{Q}_\delta^{t,k-2}(s,a) = V_\delta^{t,k-2}(s).$$

So that $\overline{Q}_\delta^{t,k} - \overline{Q}_\delta^{t,k-1} \geq 0$. We conclude that $\mathbf{P}[\overline{Q}_\delta \geq Q^*] \geq 1 - \frac{\delta}{ct^2}$. Repeating the same argument for the lower estimate, $\underline{Q}_\delta$, and applying the union bound over both and over all times completes the proof for the appropriate c. ■

The AE procedure is demonstrated in the following algorithm, which also supplies a stopping condition for sampling the model and eliminates actions when they are sub-optimal with high probability.

Input : MDP M, $\varepsilon > 0$, $\delta > 0$
Output : A policy for M
Choose arbitrarily an initial state s_0, let $t = 0$,
and let $U_0 = \{(s,a) | s \in S, a \in A\}$
repeat
  At state s_t perform any action a s.t. $(s_t, a) \in U_t$
  Receive a reward r_t, and a next state s_{t+1}
  Compute, $\overline{Q}_\delta, \underline{Q}_\delta$ from all the samples
  $t = t+1$
  $U_t = \{(s,a) | \overline{Q}_\delta(s,a) \geq \underline{V}_\delta(s)\}$
until $\forall (s,a) \in U \quad |\overline{Q}_\delta(s,a) - \underline{Q}_\delta(s,a)| < \frac{\varepsilon(1-\gamma)}{2}$;
return *Greedy*$(\underline{Q}_\delta)$

Algorithm 5: Model-Based AE Algorithm

A direct corollary from Lemma 14, is a stopping time condition to the Model-Based algorithm using the following Corollary.

Corollary 15 *[Singh and Yee (1994)] If $\tilde{Q}$ is a function such that $|\tilde{Q}(s,a) - Q^*(s,a)| \leq \varepsilon$ for all $s \in S$ and $a \in A$. Then for all s*

$$V^*(s) - V^{\tilde{\pi}}(s) \leq \frac{2\varepsilon}{1-\gamma},$$

where $\tilde{\pi} = Greedy(\tilde{Q})$.

Theorem 16 *Supposed the Model-Based AE Algorithm terminates. Then the policy, π, the algorithm returns is ε-optimal with probability at least $1-\delta$.*

Proof By Lemma 14 we know that with probability at least $1-\delta$ for every s, a and time t we have that $\underline{Q}_\delta(s,a) \leq Q^*(s,a) \leq \overline{Q}_\delta(s,a)$. Therefore, with probability of at least $1-\delta$ the optimal action has not been eliminated in any state in any time t. Furthermore, any action b in state s that has not been eliminated satisfies $Q^*(s,b) - \underline{Q}_\delta(s,b) \leq \overline{Q}_\delta(s,b) - \underline{Q}_\delta(s,b) \leq \varepsilon(1-\gamma)/2$. The result follows by Corollary 15. ■

4.2 Model-Free Learning

In this section we describe a model-free algorithm. We use two functions $\underline{Q}^t$ and $\overline{Q}^t$, which provide lower and upper estimations on Q^*, respectively. We use these functions to derive an asynchronous algorithm, which eliminates actions and supplies stopping condition. This algorithm requires space which is proportional to the space used by Q-learning and converges under the same conditions. Let us first recall the Q-learning algorithm (Watkins, 1989). The Q-learning algorithm estimates the state-action value function (for discounted return) as follows:

$$\begin{aligned} Q^0(s,a) &= 0, \\ Q^{t+1}(s,a) &= (1-\alpha_t(s,a))Q^t(s,a) + \alpha_t(s,a)(r_t(s,a) + \gamma V^t(s')), \end{aligned}$$

where s' is the state reached from state s when performing action a at time t, and $V^t(s) = \max_a Q^t(s,a)$. Set $\alpha_t(s,a) = 1/\#(s,a,t)$ for $t \in T^{s',a'}$ and 0 otherwise. [1] We define the upper estimation process as:

$$\begin{aligned}
\overline{Q}^0_\delta(s,a) &= V_{max} \ln\left(\frac{c|S||A|}{\delta}\right), \\
\overline{Q}^{t+1}_\delta(s,a) &= (1-\alpha_t(s,a))\overline{Q}^t_\delta(s,a) + \alpha_t(s,a)\Big(R(s,a) + \gamma \overline{V}^t_\delta(s') + \beta(\#(s,a,t))\Big),
\end{aligned}$$

where $c > 4$ and s' is the state reached from state s when performing action a at time t, $\overline{V}^t_\delta(s) = \max_a \overline{Q}^t_\delta(s,a)$ and the function β, which maintains the upper estimate interval is defined as:

$$\beta(k) = k\left(\sqrt{k\ln(ck^2|S||A|/\delta)} - (1-1/k)\sqrt{(k-1)\ln(c(k-1)^2|S||A|/\delta)}\right) V_{max}.$$

Analogously, we define the lower estimate $\underline{Q}_\delta$ as :

$$\begin{aligned}
\underline{Q}^0_\delta(s,a) &= -V_{max} \ln\left(\frac{c|S||A|}{\delta}\right), \\
\underline{Q}^{t+1}_\delta(s,a) &= (1-\alpha_t(s,a))\underline{Q}^t_\delta(s,a) + \alpha_t(s,a)\Big(R(s,a) + \gamma \underline{V}^t_\delta(s') - \beta(\#(s,a,t))\Big),
\end{aligned}$$

where $\underline{V}_\delta(s) = \max_a \underline{Q}_\delta(s,a)$. We claim that these processes converge almost surely to Q^*. (The proof appears in Appendix A.)

Proposition 17 *If every state-action pair is performed infinitely often then the upper (lower) estimation process, $\overline{Q}^t_\delta$ ($\underline{Q}^t_\delta$), converges to Q^* with probability one.*

The following Proposition claims that $\overline{Q}^t_\delta$ upper bounds Q^* and $\underline{Q}^t_\delta$ lower bounds Q^* with high probability.

Proposition 18 *With probability at least $1-\delta$ we have that for every state action pair (s,a) and time t:*

$$\overline{Q}^t_\delta(s,a) \geq Q^*(s,a) \geq \underline{Q}^t_\delta(s,a).$$

Proof We start by defining disjoints events such that their union is the event of $\overline{Q}$ not always being an upper bound of Q^*. Let

$$\begin{aligned}
E_{k,s,a} = \{&\text{The first time for which } \overline{Q} \text{ is not an upper bound of} \\
&Q^* \text{ is when } a \text{ is performed at state } s \text{ at the } k\text{th time}\}.
\end{aligned}$$

Note that if $\overline{Q}$ does not upper bound Q^* it implies that one of the events $E_{k,s,a}$ occurred. Next we bound the probability that an event $E_{k,s,a}$ happens. Note that the only Q value that has changed where a was performed art the kth time at state s is $\overline{Q}(s,a)$. We let t' be the time of $E_{k,s,a}$ and note that $\overline{Q}^t(s',a') \geq Q^*(s,a)$ for any $t < t'$.

$$\mathbf{P}(E_{k,s,a}) = \mathbf{P}\left(\overline{Q}^{t'}(s,a) - Q^*(s,a) < 0\right)$$

1. This particular learning rate is especially convenient, since the recurrence $X_t = (1-1/t)X_{t-1} + (1/t)\theta_t$ has the solution $X_t = (1/t)\sum_{i=1}^t \theta_i$.

Input : MDP M, $\varepsilon > 0$, $\delta > 0$
Output : A policy for M
For every state action (s,a):
$\overline{Q}(s,a) = \quad V_{max} \ln(\frac{c|S||A|}{\delta})$
$\underline{Q}(s,a) = -V_{max} \ln(\frac{c|S||A|}{\delta})$
#(s,a) = 1
Choose an arbitrary initial state s
repeat
  Let $U(s) = \{a | \overline{Q}(s,a) \geq \underline{V}(s)\}$
  choose arbitrarily action $a \in U(s)$, perform it and observe the next state s'
  $\overline{Q}(s,a) := (1 - \frac{1}{\#(s,a)})\overline{Q}(s,a) + \frac{1}{\#(s,a)}\left(R(s,a) + \gamma\overline{V}(s') + \beta(\#(s,a))\right)$
  $\underline{Q}(s,a) := (1 - \frac{1}{\#(s,a)})\underline{Q}(s,a) + \frac{1}{\#(s,a)}\left(R(s,a) + \gamma\underline{V}(s') - \beta(\#(s,a))\right)$
  $\#(s,a) := \#(s,a) + 1$; $s = s'$
until $\forall s \in S \ \forall a \in U(s) \ \ |\overline{Q}(s,a) - \underline{Q}(s,a)| < \frac{\varepsilon(1-\gamma)}{2}$;
return *Greedy*$(\underline{Q})$

Algorithm 6: Model-Free AE Algorithm

$$
\begin{aligned}
&= \mathbf{P}\left(\frac{1}{k}\sum_{i=1}^{k}(r_i + \gamma\overline{V}^{t_i}(s_i) + \beta(i)) - Q^*(s,a) < 0\right) \\
&\leq \mathbf{P}\left(\frac{1}{k}\sum_{i=1}^{k}(r_i + \gamma V^*(s_i) + \beta(i)) - Q^*(s,a) < 0\right) \\
&\leq \frac{\delta}{c|S||A|k^2},
\end{aligned}
$$

where we could apply Hoeffding's inequality since V^* is not a random variable. Now taking the union bound over all pairs (s,a) and times k completes the proof for the upper estimate. A similar argument for the lower estimate completes the proof. ■

We combine the upper and lower estimates to an algorithm, which eliminates sub-optimal actions whenever possible. Furthermore, the algorithm supplies a stopping condition that assures a near optimal policy. The model free AE algorithm is described in Algorithm 6.

A direct corollary from Proposition 18 is a stopping condition to the model free AE algorithm. The following corollary follows from Corollary 15 and its proof is similar to the proof of Theorem 16.

Corollary 19 *Suppose the Model-Free AE Algorithm terminates. Then the policy, it returns is* ε*-optimal with probability at least* $1 - \delta$.

4.3 MAB Phased Q-learning Algorithm

In contrast to previous sections concerning learning in MDPs, we restrict the setup in this section. In this limited setup we can fully exploit the connection between the MAB problem and learning

in MDPs. The setup is that of parallel sampling where the decision maker can sample every state and action pair, as opposed to the typical Q-learning setup where a single trajectory is followed. We will focus on the phased Q-learning algorithm Kearns and Singh (2002) which partitions the learning to phases. We will use a MAB black-box to perform the updates for each state and phase of the phased Q-learning algorithm. Although the parallel sampling model is not a realistic model it is often considered in theory as a relaxation of the MDP model which still captures many important aspects of the original problem; see Kearns and Singh (2002), Szepesvri and Munos (2005). The parallel sampling model can represent a situation where sampling from different states is very cheap (for example, when a simulator is available), so there is no real need to follow a single trajectory. In this case, reducing the number of samples needed for finding an optimal (or approximately optimal) policy is the main concern.

In phased Q-learning the value of $V_k(s)$ is fixed during the kth phased and updated only at the end of the phase. This implies that for every state and action (s,a) we can define a random variable $Y_s(a)$ whose value is $R(s,a)+\gamma V_k(s')$, where $R(s,a)$ is the random variable representing the reward and s' is distributed using $P^a_{s,s'}$.

Our aim is to find, at each state, the action that maximizes the expected reward, and estimate its expected reward, where the rewards are $Y_s(a)$. The phased Q-Learning can now be viewed as using the naive algorithm for the MAB problem (Algorithm 1) in order to find the best arm. In the following we show how, using more sophisticated MAB algorithms, one can improve the convergence rate of the Phased Q-Learning.

Our algorithm uses any (ε,δ)-PAC Multi-armed bandit algorithm as a black box in the learning process. In order to use the MAB algorithm B as, a black box, we define a simple interface, which requires the following procedures:

- $Init_B(\varepsilon,\delta)$ - Initialize the parameters of B.
- $GetArm_B()$ - returns the arm a that B wants to sample next.
- $Update_B(a,r)$ - informs B the latest reward r of arm a.
- $Stop_B(a,v)$ - returns TRUE if B terminates, and in such a case a is the output of B and v is its estimated value. (We assume that on termination, with probability at least $1-\delta$, the arm a is an ε-optimal arm and $|r^*-v|\leq\varepsilon$.)

The MAB Phased Q-learning algorithm uses as a black box, an algorithm B for the MAB problem. It receives as input (ε,δ) and returns a policy π which is ε-optimal with probability at least $1-\delta$.

Suppose that we have some (ε,δ)-PAC MAB algorithm B, and assume B has arm sample complexity $T_B(\varepsilon,\delta)$. Namely, with probability $1-\delta$, algorithm B terminates after at most $T_B(\varepsilon,\delta)$ and outputs a policy π which is ε-optimal. The following theorem computes the sample complexity of MAB Phased Q-Learning algorithm as a function of T_B.

Theorem 20 *Assume B is an $(\hat{\varepsilon},\hat{\delta})$-PAC multi-armed bandit algorithm. Then the MAB Phased Q-Learning(ε,δ) algorithm outputs a policy π which is ε-optimal policy with probability at least $1-\delta$, and has sample complexity of*

$$T(\varepsilon,\delta)=|S|T_B(\hat{\varepsilon},\hat{\delta})\log_\gamma(\frac{\hat{\varepsilon}}{2V_{max}})=O\left(\frac{|S|}{1-\gamma}\ln(\frac{V_{max}}{(1-\gamma)\varepsilon})T_B\left(\frac{\varepsilon(1-\gamma)^2}{2},\frac{\delta(1-\gamma)}{|S|\ln(V_{max}/\varepsilon)}\right)\right).$$

Input : $\varepsilon, \delta > 0$ and B a multi-armed bandit algorithm
Output : A policy
Let $\hat{\varepsilon} = \frac{\varepsilon(1-\gamma)^2}{2}$; $n = \log_\gamma(\hat{\varepsilon}/2V_{max}) = O(\ln(\frac{V_{max}}{(1-\gamma)\varepsilon})/(1-\gamma))$; $\hat{\delta} = \frac{\delta}{|S|n}$;
Initialize for every $s \in S$: $V_0(s) = 0$;
for $i = 1 : n$ **do**
  foreach $s \in S$ **do**
    $Init_B(\hat{\varepsilon}, \hat{\delta})$;
    repeat
      $a = GetArm_B()$;
      $(s', r) = sample(s, a)$;
      $r' = r + \gamma V_i(s')$;
      $Update_B(a, r')$;
    until $Stop(a, v) = TRUE$;
    $V_{i+1}(s) = v$; $\pi(s) = a$;
  end
end

Algorithm 7: MAB Phased Q-Learning algorithm

First we show that in each phase the norm $\|V^* - V\|_\infty$ decreases.

Lemma 21 *Assume B is an $(\hat{\varepsilon}, \hat{\delta})$-PAC multi-armed bandit algorithm, and consider the MAB Phased Q-Learning(ε, δ) algorithm using B. Then with probability at least $1 - \delta$, for all $k \leq n$, $\|V^* - V_k\|_\infty$ is bounded by $\frac{\hat{\varepsilon}}{1-\gamma} + V_{max}\gamma^k$.*

Proof First we bound the probability that B outputs an arm which is not ε-optimal. We bound the failure probability by using the union bound on all the invocations of B. There are

$$|S|n = |S| \log_\gamma(\hat{\varepsilon}/V_{max}) = O\left(\frac{|S| \ln(\frac{V_{max}}{(1-\gamma)\varepsilon})}{1-\gamma}\right)$$

initializations of algorithm B and for each invocation the failure probability is bounded by $\delta/|S|n$. Thus, the failure probability is at most δ.

Next, we show that the error contracts in every phase. We compare the value vector, V_k, with the standard value iteration value vector $\hat{V}_k$ for the case of a known model (at the end of the k-th step). Formally,

$$\hat{V}_{k+1}(s) = \max_u \{\mathbb{E}[R(s,u)] + \gamma \mathbb{E}_{s'}[\hat{V}_k(s')]\},$$

where s' is distributed according to $P^u_{s,s'}$ and $\hat{V}_0 = 0$.

We show by induction on the number of phases, that $d_k = \|V_k - \hat{V}_k\|_\infty \leq \frac{\hat{\varepsilon}}{1-\gamma}$. The base of the induction, $t = 0$, for every state s we have $d_0 = |V_0(s) - \hat{V}_0(s)| = 0$. We assume that the induction assumption holds for $t < k$ and prove for k. Let $m_{s,a}$ denote the number of times the state action pair (s,a) was sampled in the k-th iteration.

$$|V_k(s) - \hat{V}_k(s)| \quad = \quad \left| \max_u [\frac{1}{m_{s,u}} \sum_{i=1}^{m_{s,u}} r(s,u) + \gamma V_{k-1}(s'_i)]\right.$$

$$
\begin{aligned}
& \left. - \max_a [\mathbb{E}[R(s,a)] + \gamma \sum_{s'} P^a_{s,s'} \hat{V}_{k-1}(s')] \right| \\
\leq & \max_{\rho \in \{-\hat{\varepsilon}, \hat{\varepsilon}\}} \left| \max_u [\mathbb{E}[R(s,u)] + \gamma \sum_{s'} P^u_{s,s'} V_{k-1}(s')] + \rho \right. \\
& \left. - \max_a [\mathbb{E}[R(s,a)] + \gamma \sum_{s'} P^a_{s,s'} \hat{V}_{k-1}(s')] \right| \\
\leq & \hat{\varepsilon} + \max_a \left| \gamma \sum_{s'} P^a_{s,s'} (V_{k-1}(s') - \hat{V}_{k-1}(s')) \right| \\
\leq & \hat{\varepsilon} + \gamma d_{k-1} \\
\leq & \hat{\varepsilon} + \gamma (\frac{\hat{\varepsilon}}{1-\gamma}) = \frac{\hat{\varepsilon}}{1-\gamma}.
\end{aligned}
$$

To conclude the proof note that for the value iteration we have that $\|\hat{V}_k - V^*\|_\infty \leq \gamma^k V_{max}$, where $\hat{V}_0 = 0$ (see, e.g., Bertsekas and Tsitsiklis, 1995). ■

Lemma 22 *When the MAB Phased Q-Learning algorithm terminates, the policy π it returns is ε-optimal with probability at least $1-\delta$.*

Proof By Lemma 21 we have that with probability at least $1-\delta$ the difference $\|V_k - V^*\|_\infty \leq \frac{\hat{\varepsilon}}{1-\gamma} + V_{max}\gamma^k$. Since $\hat{\varepsilon} = \varepsilon(1-\gamma)^2/2$, we have that $\|V_k - V^*\|_\infty \leq \varepsilon(1-\gamma)/2 + V_{max}\gamma^k$. The lemma follows from our choice of $n = \log_\gamma(\varepsilon(1-\gamma)/2V_{max})$. ■

We can now complete the proof Theorem 20.

Proof The correctness follows from Lemma 22. We bound the sample complexity as follows. By definition, the MAB Phased Q-Learning algorithm samples at each state and action during every phase $T_B(\hat{\varepsilon}, \hat{\delta})$. By definition of the algorithm, the number of phases is $n = O(\ln(V_{max}/\hat{\varepsilon})/(1-\gamma))$, and each phase is composed from $|S|$ MAB instances. This completes the bound on the sample complexity. ■

Applying the multi-armed bandit algorithms described in the previous sections we derive the following corollary. We show that by using the *median elimination* algorithm, the arm sample complexity can be reduced by a factor of $\log(|A|)$.

Corollary 23 *Let B be the median elimination algorithm. MAB Phased Q-Learning algorithm has sample complexity*

$$
T(\varepsilon, \delta) = O\left(\frac{|S|\ |A| V_{max}^2}{(1-\gamma)^5 \varepsilon^2} \ln(\frac{V_{max}}{(1-\gamma)\varepsilon}) \ln(\frac{|S| \ln(V_{max}/\varepsilon)}{\delta(1-\gamma)}) \right).
$$

Next we introduce an almost matching lower bound. Let us introduce some more notation before we proceed. Let T denote the time until an RL algorithm stops (this may be in general a random number). For a given RL algorithm L and a given MDP L we denote by $\mathbb{E}^{L,M}$ the expectation with respect to randomization in both the algorithm and the MDP.

Theorem 24 *Let L be a learning algorithm for MDPs under the parallel sampling model. There are constants $C_1, C_2, \varepsilon_0, \delta_0, \gamma_0$ such that for every $\varepsilon \in (0, \varepsilon_0)$, $\delta \in (0, \delta_0)$, and $\gamma \in (0, \gamma_0)$ if L returns an ε-optimal policy with probability of at least $1-\delta$ for every MDP with discount factor γ there exist an MDP M for which:*

$$\mathbb{E}^{L,M}[T] \geq C_1 \frac{|S|\,|A|}{(1-\gamma)^2\varepsilon^2} \log\left(\frac{C_2}{\delta}\right) = \Omega\left(\frac{|S|\,|A|}{(1-\gamma)^2\varepsilon^2} \log(\frac{1}{\delta})\right) .$$

Proof Consider the following construction which was used in Theorem 1 of Mannor and Tsitsiklis (2004) for the MAB problem. A MAB problem with $|A|$ arms is given. We enumerate the arms from 0 to $|A|-1$, and fix $\hat{\varepsilon} > 0$. In each of the states one of the following hypotheses is true:

$$H_0 : r(0) = 1/2 + \hat{\varepsilon}; \qquad r(i) = 1/2 \quad (i = 1, 2, \ldots, |A-1|),$$

and for $\ell = 1, 2, \ldots, |A|-1$:

$$H_\ell : r(0) = 1/2 + \hat{\varepsilon}; \qquad r(i) = 1/2 \quad (i = 1, 2, \ldots, |A-1|, i \neq \ell); \quad r(\ell) = 1/2 + 2\hat{\varepsilon}.$$

Let $\mathbb{E}_\ell$ be the expectation given than hypothesis H_ℓ is true, and $A(s)$ the event that the algorithm errs in state s. In Lemma 4 in Mannor and Tsitsiklis (2004) it was proved that there are constants c_1 and c_2 such that for $\hat{\varepsilon} < \hat{\varepsilon}_0$ every algorithm that can identify the true hypothesis with probability $1-\hat{\delta}$ for all the hypotheses must satisfy that:

$$\mathbb{E}_0[T] \geq c_1 \frac{|A|-1}{\hat{\varepsilon}^2} \log(\frac{c_2}{\hat{\delta}}). \tag{4}$$

We create the following set of MDPs. In each possible MDP there are $|S|$ states and $|A|$ actions in each state. All the states are absorbing and have one of the above A hypotheses per state. The reward in each state behaves according to H_0 or one of the H_ℓ. (There are $|A|^{|S|}$ possible MDPs.) We set $\hat{\varepsilon} = 2(1-\gamma)\varepsilon$. We run algorithm L until termination. When L terminates it returns a policy which is ε-optimal with probability of at least $1-\delta$. Since every state is absorbing, and by our choice of $\hat{\varepsilon}$, it implies that the right hypothesis was found in all states. Note that even if L returns a randomized policy, we will determine that the action with the highest reward is the best one (this is the reason for the factor of 2 in determining $\hat{\varepsilon}$). By taking the sum of Eq. (4) over all states we obtain that

$$\mathbb{E}^{L,M}[T] \geq c_1 \frac{|A|-1}{\hat{\varepsilon}^2} |S| \log(\frac{c_2}{\delta}).$$

The result follows by an appropriate choice of constants. ■

5. Experiments

In this section we show four types of MDPs in which the number of samples used by AE procedures is significantly smaller than the number of samples used by standard Q-learning and ε-greedy Q-learning. Both model free AE algorithm and standard Q-learning choose the action in each state uniformly at random. In our experiments we focused on the steady state norm (L_1 weighted by steady state probabilities) rather than the L_∞ norm to emphasize the average behavior. We note that we use the steady state rather than the discounted steady state. We run AE Q-learning algorithm from Section 4.2 with the same input (for actions that were not eliminated) as a standard Q-learning algorithm. The following experiments were conducted:

1. **A queueing system.** The MDP represents a queueing problem that appears in Differentiated Services (Aiello et al., 2000, Kesselman et al., 2004). The basic settings are that the arriving packets have different values and they are buffered in a FIFO queue before being sent. The major constraints are that we reject or accept a packet upon its arrival (no preemption) and that the buffer has limited capacity. We have analyzed a queue of size five and three different packets values, $1, 20, 150$. In each time unit we either receive a packet or send a packet according to some distribution. We modeled the queueing problem via a discounted MDP with discount factor $\gamma = 0.99$. The AE model-free algorithm[2] was compared with ε-greedy Q-learning with epsilon varying from 0.05 to 0.2. In Figure 1 we present the results for ε which was empirically best, $\varepsilon = 0.1$. In this experiment we used a fixed step size. We focused here on the fraction of times in which optimal actions were performed and on the value function criterion. The results are demonstrated in Figure 1, in which we see that not only AE has better results but the variance in the results is much smaller in both the fraction of times that almost optimal actions were performed and in the value function. Figure 2 demonstrates the elimination rate of the AE procedure.

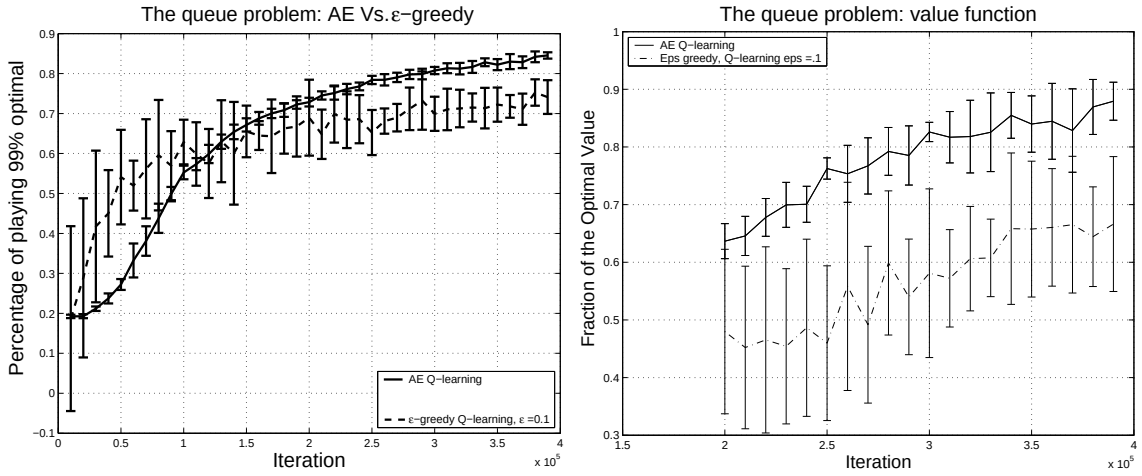

Figure 1: Example of a Queue of size 5 with three types of packets with values $1, 20, 150$. The discount factor is set to 0.99. We disregard the full queue state in which every action is optimal. We repeated each experiment 15 times and the error bars represent 1 standard deviation.

2. **Random MDPs.** Two types of random MDPs were randomly generated. In both types there were 20 states and 50 actions in each state. The first type is due to Puterman (1994) and is a sparse MDP, such that each action can reach only three states. The second type of random MDPs is dense, such that the next state distribution is randomly chosen for each state-action pair and might include all states. For both MDPs the immediate reward expectation is randomly chosen in the interval $[0, 10]$. Results of ten runs are presented by Figure 3 for the

2. Since we were interested in the short term results rather than the long term, we initialized the upper and lower values to similar values and allowed elimination only after an exploration period, we still used the β function for both the upper and lower estimates as stated in the theoretical part up to constants.

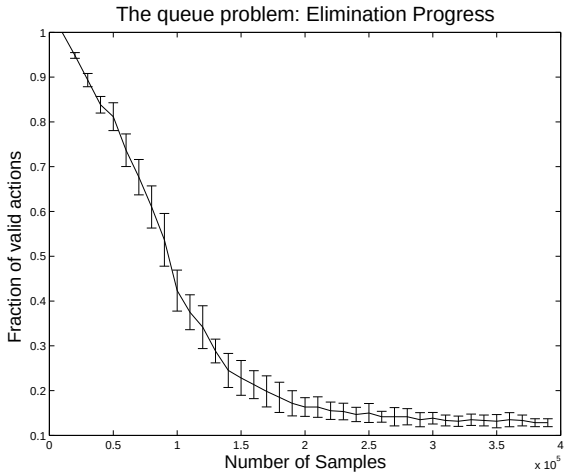

Figure 2: Example of a Queue of size 5 with three types of packets with values $1, 20, 150$. The discount factor is set to 0.99. This figure demonstrates the elimination rate. We repeated each experiment 15 times and the error bars represent 1 standard deviation.

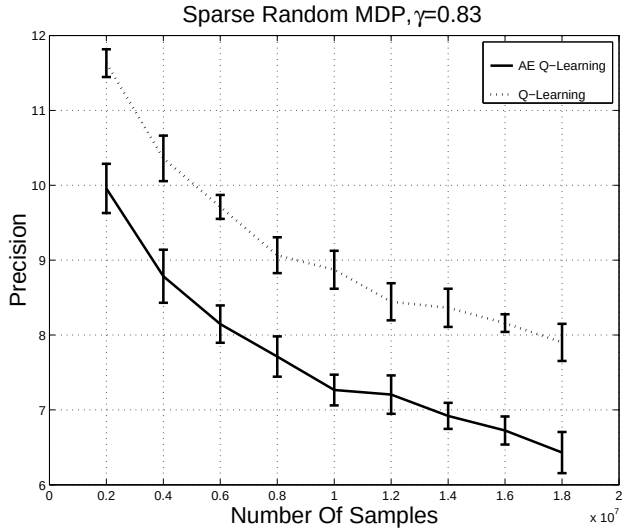

Figure 3: Example of a 20 state sparse randomly generated MDPs with 50 actions in each state, where $\gamma = 0.833$ (as in Puterman (1994).) The precision is the distance of the Q-function from the optimal Q-function. We repeated each experiment 10 times and the error bars represent 1 standard deviation.

sparse MDP, in this experiment the model free AE algorithm needs only about half the samples used by the Q-learning to achieve the same precision. The precision is measured as the distance of the Q-function from the optimal function in steady state norm. In Figure 4 for dense MDP, the results are similar. The AE algorithm required about 40% fewer samples.

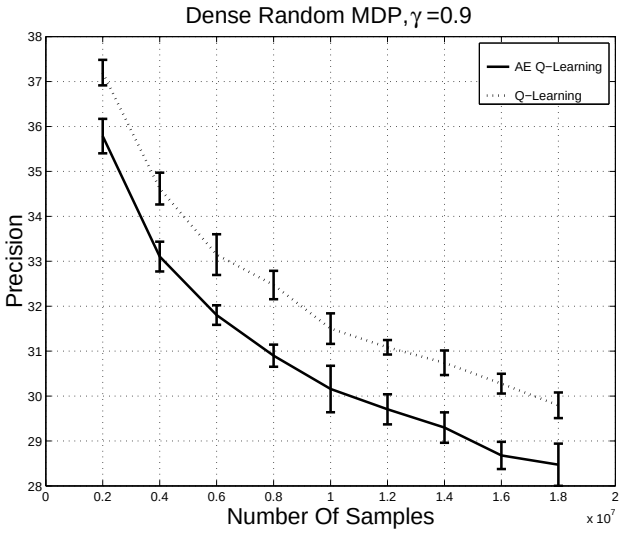

Figure 4: Example of a 20 state dense randomly generated MDPs with 50 actions in each state, $\gamma = 0.9$. The error bars represent 1 standard deviation.

3. **Howard's automobile replacement problem.** This MDP represents another realistic problem—Howard's automobile replacement problem Howard (1960). This problem contains 40 states, in each state there are 41 actions. See Howard (1960) for a detailed description. This problem was considered as a benchmark by several authors in the optimization community. We used the model free AE algorithm for this problem with discount factor $\gamma = 0.833$ against standard Q-learning and the results appear in Figure 5. A significant improvement is evident.

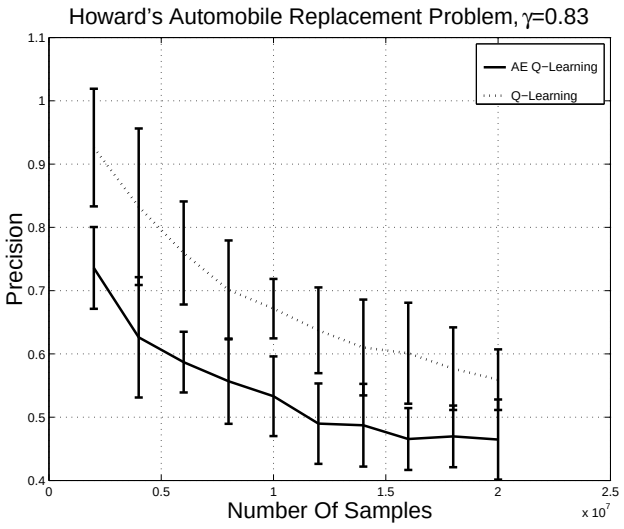

Figure 5: Example of Howard's Automobile Replacement Problem, where the discount factor, γ, is 0.833. The norm is the steady state norm. The error bars represent 1 standard deviation.

6. Future Directions

Extending the concept of action elimination to large state spaces is probably the most important direction. The extension to function approximation, which approximates the value function, requires some assumptions on the value (or Q) function approximation architecture. Following Kakade and Langford (2002) we can consider value functions that can be approximated under the infinity norm. For an example of such an algorithm see (Ormoneit and Sen (2002)). If convergence rate of the function approximation is provided, as in (Ormoneit and Sen (2002)), then an AE procedure can be derived as before.

Acknowledgements

E.E. and Y.M. was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, by a grants no. 525/00 and 1079/04 from the Israel Science Foundation and an IBM faculty award. The work was done while E.E was a graduate student at Tel Aviv University. S.M. was partially supported by the Natural Sciences and Engineering Research Council of Canada and by the Canada Research Chairs Program. We thank Alex Strehl and Csaba Szepesvari for helpful discussions. This publication only reflects the authors' views.

Appendix A. Proof of Proposition 17

In order to show the almost sure convergence of the upper and lower estimations, we follow the proof of Bertsekas and Tsitsiklis (1996). We consider a general type of *iterative stochastic algorithms*, which is performed as follows:

$$X_{t+1}(i) = (1-\alpha_t(i))X_t(i) + \alpha_t(i)((HX_t)(i) + w_t(i) + u_t(i)),$$

where w_t is a bounded random variable with zero expectation and each H is a pseudo contraction mapping (See Bertsekas and Tsitsiklis, 1996, for details).

Definition 25 *An iterative stochastic algorithm is well behaved if:*

1. *The step size $\alpha_t(i)$ satisfies (1) $\sum_{t=0}^{\infty}\alpha_t(i) = \infty$, (2) $\sum_{t=0}^{\infty}\alpha_t^2(i) < \infty$ and (3) $\alpha_t(i) \in (0,1)$.*
2. *There exists a constant A that bounds $w_t(i)$ for any history F_t, i.e., $\forall t,i:\ |w_t(i)| \le A$.*
3. *There exists $\gamma \in [0,1)$ and a vector X^* such that for any X we have $||HX - X^*|| \le \gamma||X - X^*||$, where $||\cdot||$ is any norm.*
4. *There exists a nonnegative random sequence θ_t, that converges to zero with probability 1, and is such that*
$$\forall i,t \quad |u_t(i)| \le \theta_t(||X_t|| + 1)$$

We first note that the Q-learning algorithm satisfies the first three criteria and the fourth criteria holds trivially since $u_t = 0$, thus its convergence follows if all state-action pairs are tried infinitely often (see Proposition 5.6 in Bertsekas and Tsitsiklis, 1996). The upper estimate has an additional noise term, u_t. If we show that it satisfies the fourth requirement, then the convergence will follow.

Lemma 26 *The upper estimation algorithm is well behaved.*

Proof In the convergence proof of Q-learning, it was shown that requirements 1–3 are satisfied, this implies that the upper estimates satisfies them as well. Now we let $u_t = \theta_t = c\sqrt{\frac{\ln(\#(s,a,t))}{\#(s,a,t)}}V_{max}$. It follows that θ_t converges to zero, thus

$$|u_t(i)| = \theta_t \leq \theta_t(||X_t|| + 1).$$

■

Similar result holds for the lower estimate as well.

References

W. A. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosen. Competitive queue policies for differentiated services. In *INFOCOM*, 2000. (To appear in J. of Algorithms).

D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18:155–193, 1979.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proc. 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, 1995.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM J. on Computing*, 32(1):48–77, 2002.

D. A. Berry and B. Fristedt. *Bandit Problems*. Chapman and Hall, 1985.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1995.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

V. S. Borkar and S.P Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control Optim.*, 38(2):447–469, 2000.

E. Even-Dar and Y. Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5:1–25, 2003. (A preliminary version appeared in the Fourteenth Annual Conference on Computation Learning Theory (2001), 589-604.).

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

R. Howard. *Dynamic programming and Markov decision processes*. MIT press, 1960.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274. Morgan Kaufmann, 2002.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002. (A preliminary version appeared in ICML (1998), 260-268.).

M. Kearns and S. P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Neural Information Processing Systems 10*, pages 996–1002, 1998.

A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM J. on Computing*, 33(3):563–583, 2004. (A preliminary version appeared in ACM Symposium on Theory of Computing (2001), 520-529.).

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

J. MacQueen. A modified dynamic programming method for Markov decision problems. *J. Math. Anal. Appl.*, 14:38–43, 1966.

S. Mannor and J. N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004. (A preliminary version appeared in the Sixteenth Annual Conference on Computation Learning Theory (2003), 418-432.).

D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

M. Puterman. *Markov Decision Processes*. Wiley-Interscience, 1994.

H. Robbins. Some aspects of sequential design of experiments. *Bull. Amer. Math. Soc.*, 55:527–535, 1952.

S. P. Singh and R. C. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

R. Sutton and A. Barto. *Reinforcement Learning*. 1998.

Cs. Szepesvri and R. Munos. Finite time bounds for sampling based fitted value iteration. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, page 881886, 2005.

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.