

PAC Bounds for Multi-Armed Bandit and Markov Decision Processes

Eyal Even-Dar¹, Shie Mannor², and Yishay Mansour¹

¹ School of Computer Science
Tel Aviv University, Tel-Aviv 69978, Israel
{evend,mansour}@cs.tau.ac.il

² Department of Electrical Engineering
Technion, Haifa 32000, Israel
shie@tx.technion.ac.il

Abstract. The bandit problem is revisited and considered under the PAC model. Our main contribution in this part is to show that given n arms, it suffices to pull the arms $O(\frac{n}{\epsilon^2} \log \frac{1}{\delta})$ times to find an ϵ -optimal arm with probability of at least $1 - \delta$. This is in contrast to the naive bound of $O(\frac{n}{\epsilon^2} \log \frac{n}{\delta})$. We derive another algorithm whose complexity depends on the specific setting of the rewards, rather than the worst case setting. We also provide a matching lower bound.

We show how given an algorithm for the PAC model Multi-Armed Bandit problem, one can derive a batch learning algorithm for Markov Decision Processes. This is done essentially by simulating Value Iteration, and in each iteration invoking the multi-armed bandit algorithm. Using our PAC algorithm for the multi-armed bandit problem we improve the dependence on the number of actions.

1 Introduction

The Multi-armed bandit problem is one of the classical problems in decision theory. The problem is very simple to model - there are a number of alternative arms, each with a stochastic reward with initially unknown expectation, and our aim is to maximize the sum of rewards, which in this setting is to choose the arm with the highest expected reward. One of the attractive features of the multi-armed bandit problem, is that although its simplicity, it already encompasses many of 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 [18] 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 [4] for a review of the classical results on the multi-armed bandit problem.)

Two notable concepts are the Gittins index and efficiency of allocation rules. In the seminal work of Gittins and Jones [12] a simple optimal decision rule was derived, this rule is known as the Gittins Index. The model in question is Bayesian, where the a-priory distribution of each arm is uniform and reward is discounted. The Gittins index is a way to quantify how good is an arm expected to be, and the derived policy is to simply choose the arm with the highest index. For the expected regret model, where the regret is defined as the difference between the optimal return and the online algorithm return, the seminal work of Lai and Robbins [17] 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 [17] were shown to be efficient, in the sense that the convergence rates are optimal. The adversarial multi-armed bandit problem was considered in [2, 3], 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 are concerned with PAC style bounds. Our 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 we need to choose one specific arm, and we are 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.

Before we start describing our results, we would like to point out that there is a very simple strategy for the PAC multi-armed bandit problem. Namely, sample each arm of the n arms for $O(1/\epsilon^2 \log(n/\delta))$ times, compute the average reward of each arm, and return the arm with the highest average. Our results improve on this simple bound in two different ways, using two different algorithms.

The first algorithm, *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 $O(n/\epsilon^2 \log(1/\delta))$, which improves the naive bound by a factor of $\log n$. We also show lower bounds, which are tight for $\delta \geq \frac{1}{n}$.

After deriving our PAC algorithm for the multi-armed bandit problem, we establish an interesting connection between learning in Markov Decision Processes (MDPs) and PAC algorithms for multi-armed bandit problem. We show how to use, as a black box, any PAC algorithm for the multi-armed bandit problem, and derive a learning algorithm for MDPs. The derived algorithm is a modification of the Phased Q-Learning algorithm of Kearns and Singh [15]. The basic idea is that we can view each phase of the algorithm as a separate multi-armed bandit problem, at each state. The PAC nature of our algorithm ensures us that a near optimal action is eventuated quickly. Using our general transformation we are

able to derive an algorithm with a better dependency on the number of actions, when using our median elimination algorithm.

To put our work in perspective, most of the theoretical work concerning convergence of Reinforcement Learning algorithms concentrated on the performance in the limit. Specifically, it was shown that the popular Q-learning algorithm ([9]) converges in the limit to the optimal Q-function. This result was extended to more general settings in [5, 19, 6] and other algorithms as well (see [5] for a review). Most of the research is focused on finding the optimal policy (rather than ϵ -optimal policy), and the arguments are usually based on the stochastic approximation algorithm and the Ordinary Differential Equation (ODE) method. Consequently, the results are usually in the limit, and finite sample bounds are not provided (c.f., [6]).

In recent years there has been interest in applying PAC style analysis to MDPs, obtaining PAC style bounds, and deriving provably efficient algorithms. To name some of the contributions along this new research direction: E^3 that efficiently computes an ϵ -optimal policy [15], sparse sampling [16], restricted class of strategies [13], convergence rate for Q-learning [10], average reward games [7], model based learning [14], and adaptive control of linear systems [11]. Our work concentrates on the basic question of how many samples are needed for finding an ϵ -optimal policy with probability $1 - \delta$. In addition, our derived algorithm is computationally efficient.

The paper is organized as follows. Section 2 specifies the multi-armed bandit and the MDP models. Algorithms for finding the almost best arm with high probability are presented and analyzed in Section 3. Lower bounds on the minimal number of samples needed to find an almost optimal arm are provided in Section 4. The phased Q-learning is described in Section 5.

2 The Models

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 if the reward is bounded in $[0, 1]$). Denote the arms by a_1, \dots, a_n and $p_i = E[R(a_i)]$. For simplicity of notations we enumerate the arms according to their expected reward $p_1 > p_2 > \dots > p_n$.

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., $E[R(a)] \geq r^* - \epsilon$.

An algorithm for the multi armed bandit 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 $t - 1$.

An algorithm is a (ϵ, δ) -PAC algorithm for the multi armed bandit if it outputs an ϵ -optimal arm, a' , with probability at least $1 - \delta$, when it terminates. The number of time steps the algorithm performs until it terminates is called the *arm sample complexity* of the algorithm.

Markov Decision Processes: We define a Markov Decision process (MDP) as follows.

Definition 1. 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, $P_{i,j}^a(M)$ is the transition probability from state i to state j when performing action $a \in A$ in state i , and $R_M(s, a)$ is the reward distribution when performing action a in state s .

A strategy for an MDP assigns, at each time $t+1$, for each state s a probability for performing action $a \in A$, given a history $F_t = \{s_1, a_1, r_1, \dots, s_t, a_t, r_t\}$ which includes the states, actions and rewards observed until time t . The action chosen by strategy π , at time t is denoted by a_t . The state at time t is denoted by s_t and similarly the reward obtained at time t is r_t (distributed according to $R_M(s_t, a_t)$), and the next state s_{t+1} (distributed according to $P_{s_t, s_{t+1}}^{a_t}(M)$). We combine the sequence of rewards to a single value called *return*, and our goal is to maximize the return. In this work we focus on *discounted return*, with discount factor $\gamma \in (0, 1)$. The discounted return of policy π is $V_M^\pi = \sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward observed at time t .

We assume that $R_M(s, a)$ is non-negative and bounded by R_{max} , i.e., $0 \leq R_M(s, a) \leq R_{max}$ for every s and a . This implies that the discounted return is bounded by $V_{max} = \frac{R_{max}}{1-\gamma}$.

We define a value function for each state s , under policy π , as $V_M^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i]$, where the expectation is over a run of policy π starting at state s . We further define the Q-function for a state-action pair as the expected reward from state s given that a was played and π is played from that time on $Q_M^\pi(s, a) = E[R_M(s, a)] + \gamma \sum_{s'} P_{s, s'}^a(M) V_M^\pi(s')$.

Let π^* be an optimal policy which maximizes the return from any start state. This implies that for every policy π and every state s we have $V_M^{\pi^*}(s) \geq V_M^\pi(s)$, and $\pi^*(s) = \operatorname{argmax}_a (E[R_M(s, a)] + \gamma (\sum_{s'} P_{s, s'}^a(M) V_M^{\pi^*}(s')))$. We use V^* for $V_M^{\pi^*}$. We say that a policy π is ϵ -optimal if $\|V_M^* - V_M^\pi\|_\infty \leq \epsilon$.

We use a Sampling Model, which has an invocation $sample(s, a)$ that returns (s', r) where the next state s' , distributed according to $P_{s, s'}^a(M)$ and a reward r distributed according to $R_M(s, a)$.

3 PAC Bounds for Multi Armed Bandit Problems

In this section we look for an (ϵ, δ) -PAC algorithms for the multi-armed bandit problem. Such algorithms are required to output with probability $1 - \delta$ an ϵ -optimal arm. A naive solution samples each arm $O(\frac{1}{(\epsilon/2)^2} \log(\frac{n}{\delta}))$, so that with probability $1 - \frac{\delta}{n}$, for every arm we approximate the expected reward within $\epsilon/2$.

Algorithm : **Naive**(ϵ, δ)

1. For every arm $a \in A$: Sample it $\ell = \frac{4}{\epsilon^2} \log(\frac{2n}{\delta})$ times.
2. Let \hat{p}_a be the average reward of arm a .
3. output $a' = \arg \max_{a \in A} \{\hat{p}_a\}$.

Theorem 1. *The algorithm Naive(ϵ, δ) is an (ϵ, δ) -PAC algorithm with arm sample complexity $O(\frac{n}{\epsilon^2} \log(\frac{n}{\delta}))$.*

Proof. Let a' be an arm for which $E(R(a')) < r^* - \epsilon$. We want to bound the probability of the event $\hat{p}_{a'} > \hat{p}_{a^*}$.

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

where the last inequality uses the Hoeffding inequality. Choosing $\ell = \frac{4}{\epsilon^2} \log(\frac{2n}{\delta})$ assures that $P(\hat{p}_{a'} > \hat{p}_{a^*}) \leq \frac{\delta}{n}$. The sample complexity is immediate from the definition of the algorithm, which completes the proof. \square

3.1 Successive Elimination

To motivate the successive elimination algorithm, suppose 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 $\frac{1}{\Delta_i^2} \log(\frac{n}{\delta})$ times, and then eliminate it. This is done in phases. Initially, we sample each arm $\frac{1}{\Delta_n^2} \log(\frac{n}{\delta})$ 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 $O((\frac{1}{\Delta_{n-i}^2} - \frac{1}{\Delta_{n-i+1}^2}) \log(\frac{n}{\delta}))$ times and then eliminate the empirically worst arm.

Algorithm : **Successive Elimination Known Biases** (δ)

1. Set $S = A$; $t_i = \lceil 4/\Delta_i^2 \log(n/\delta) \rceil$; and $t_{n+1} = 0$.
2. Set for every arm a : $\hat{p}_a = 0$.
3. For $i = 0$ to $n - 2$ DO
 - (a) Sample every arm $a \in S$ for $t_{n-i} - t_{n-i+1}$ times.
 - (b) Let \hat{p}_a be the average reward of arm a (in all rounds).
 - (c) Set $S = S \setminus a_{\min}$, where $a_{\min} = \arg \min_{a \in S} \{\hat{p}_a\}$.
4. output S .

Theorem 2. *The Successive Elimination with Known Biases algorithm is an $(0, \delta)$ -PAC algorithm and its arm sample complexity is $O\left(\log(\frac{n}{\delta}) \sum_{i=2}^n \frac{1}{\Delta_i^2}\right)$.*

Proof. We claim that this algorithm outputs the best arm with probability $1 - \delta$. This is done by showing that, in each phase, the probability of eliminating the best arm is bounded by $\frac{\delta}{n}$. It is clear that the failure probability at phase i is

maximized if all the $i-1$ worst arms have been eliminated in the first $i-1$ phases. Since we eliminate a single arm at each phase of the algorithm, the probability of the best arm being eliminated at phase i is bounded by

$$Pr[\hat{p}_1 < \hat{p}_2, \hat{p}_1 < \hat{p}_3, \dots, \hat{p}_1 < \hat{p}_{n-i}] \leq Pr[\hat{p}_1 < \hat{p}_{n-i}]$$

The probability that $\hat{p}_1 < \hat{p}_{n-i}$, after sampling each arm for $O(\frac{1}{\Delta_i^2} \log \frac{n}{\delta})$ times, is bounded by $\frac{\delta}{n}$. Therefore, the total probability of failure is bounded by δ and the arm sample complexity is,

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

□

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.

Algorithm : **Successive Elimination**(δ)

1. Set $t = 1$ and $S = A$
2. Set for every arm a : $\hat{p}_a^1 = 0$.
3. Sample every arm $a \in S$ once and let \hat{p}_a^t be the average reward of arm a by time t .
4. Let $\hat{p}_{max}^t = \max_{a \in S} \hat{p}_a^t$ and $\alpha_t = \sqrt{\frac{\log(cnt^2/\delta)}{t}}$
5. For every arm $a \in S$ such that $\hat{p}_{max}^t - \hat{p}_a^t \geq 2\alpha_t$ set $S = S \setminus a$
6. $t = t + 1$.
7. If $|S| > 1$ Then Go to 3 Else output S .

Theorem 3. *The Successive Elimination algorithm is a $(0, \delta)$ -PAC algorithm, and with probability at least $1 - \delta$ its arm sample complexity is bounded by $O\left(\sum_{i=2}^n \frac{\log(\frac{n}{\delta \Delta_i})}{\Delta_i^2}\right)$.*

Proof. Our main argument is that 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,

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

This implies 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

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

The arm sample complexity is $t_2 + \sum_{i=2}^n t_i$, which completes the proof. \square

Remark 1. We can improve the dependence on the parameter Δ_i if at the t -th phase we sample each action in S_t for 2^t times rather than once and take $\alpha_t = \sqrt{\frac{\log(cn \log(t)/\delta)}{t}}$. This will give us a bound on the number of samples with a dependency of

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

Remark 2. 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 $O(\frac{1}{\epsilon^2} \log(\frac{k}{\delta}))$. 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

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

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

3.2 Median elimination

The following algorithm uses a clever trick for substituting $\log(1/\delta)$ for $\log(n/\delta)$ of the naive bound. The idea is to throw 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.

Algorithm : **Median Elimination**(ϵ, δ)

1. Set $S = A$.
2. $\epsilon_1 = \epsilon/4, \delta_1 = \delta/2, \ell = 1$.
3. Sample every arm $a \in S$ for $\frac{1}{(\epsilon_\ell/2)^2} \ln(3/\delta_\ell)$ times, and let \hat{p}_a^ℓ denote its empirical value.
4. Find the median of \hat{p}_a^ℓ , denoted by m_ℓ .
5. $S_{\ell+1} = S_\ell \setminus \{a : \hat{p}_a^\ell < m_\ell\}$.
6. If $|S_\ell| = 1$ Then output S_ℓ ,
Else $\epsilon_{\ell+1} = \frac{3}{4}\epsilon_\ell; \delta_{\ell+1} = \delta_\ell/2; \ell = \ell + 1; \text{Go to 3.}$

Theorem 4. *The Median Elimination(ϵ, δ) algorithm is an (ϵ, δ) -PAC algorithm and its arm sample complexity is $O(\frac{n \ln(\frac{1}{\delta})}{\epsilon^2})$.*

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 1. *For the Median Elimination(ϵ, δ) algorithm we have that*

$$Pr[\max_{j \in S_\ell} p_j \leq \max_{i \in S_{\ell+1}} p_i + \epsilon_\ell] \geq 1 - \delta_\ell$$

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 $E_1 = \{\hat{p}_1 < p_1 - \epsilon_1/2\}$, which is the case that the empirical estimate of the best arm is pessimistic. By step 3 of the algorithm, we sample sufficiently such that $Pr[E_1] \leq \delta_1/3$.

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

$$Pr[\hat{p}_j \geq \hat{p}_1 \mid \hat{p}_1 \geq p_1 - \epsilon_1/2] \leq Pr[\hat{p}_j \geq p_j + \epsilon_1/2 \mid \hat{p}_1 \geq p_1 - \epsilon_1/2] \leq \delta_1/3$$

Let #bad be the number of arms which are not ϵ_1 -optimal but are empirically better than the best arm. We have that $E[\text{\#bad} \mid \hat{p}_1 \geq p_1 - \epsilon_1/2] \leq n\delta_1/3$. Next we apply Markov inequality to obtain,

$$Pr[\text{\#bad} \geq n/2 \mid \hat{p}_1 \geq p_1 - \epsilon_1/2] \leq \frac{n\delta_1/3}{n/2} = 2\delta_1/3.$$

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

Next we prove that arm sample complexity is bounded by $O(\frac{n \ln(\frac{1}{\delta})}{\epsilon^2})$.

Lemma 2. *The Median Elimination(ϵ, δ) has $O(\frac{n \ln(\frac{1}{\delta})}{\epsilon^2})$ arm sample complexity.*

Proof. The number of arm samples in the ℓ -th round is $\frac{4n_\ell \ln(3/\delta_\ell)}{\epsilon_\ell^2}$. 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. $\epsilon_1 = \epsilon/4$; $\epsilon_\ell = \frac{3}{4}\epsilon_{\ell-1} = (\frac{3}{4})^{\ell-1} \epsilon/4$

Therefore we have

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

□

Now we can prove Theorem 4.

Proof. From Lemma 2 we have that the sample complexity is bounded by $O(\frac{n \ln(1/\delta)}{\epsilon^2})$. By Lemma 1 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 $\sum_{i=1}^{\log_2(n)} \epsilon_i \leq \epsilon$. □

4 A Lower Bound

In this section we provide a lower bound for the arm sample complexity. We do this by showing that one can convert an algorithm for solving an n -armed bandit problem with sample complexity T , to an algorithm that determines the bias of a single coin in expected time $O(T/n)$. First we define the coin bias problem (e.g., [1]).

Definition 2. *The input to a coin bias problem is a binary random variable X , such that $P[X = 1] \in \{1/2 + \epsilon, 1/2 - \epsilon\}$. The aim is to determine, with probability at least $1 - \delta$, the correct bias of the coin (i.e., the value of $P[X = 1]$), given access to i.i.d. samples of X . The sample complexity of an (ϵ, δ) -PAC algorithm is the maximum over $P[X = 1] \in \{1/2 + \epsilon, 1/2 - \epsilon\}$ of the expected number of samples of the random variable X that the algorithm makes.*

Lemma 3. *Given an (ϵ, δ) -PAC multi-armed bandit algorithm A for n -armed bandit problem that has expected sample complexity T , there exists an $(\epsilon, 2\delta + 1/n)$ -PAC algorithm for the coin bias problem with expected sample complexity $O(T/n)$.*

Proof. Let C_1, \dots, C_n be i.i.d. $\{0, 1\}$ valued random variables, such that $P(C_i = 1) = 1/2 - \epsilon$. Let $I(X, i)$ represent n random variables, Y_1, \dots, Y_n , where $Y_i = X$ and $Y_k = C_k$, for $k \neq i$.

Given as an input a random variable X , we run two copies of A , the first called **min** with $I(X, i)$ and the second called **max** with $I(1 - X, j)$, where i and j are chosen at random uniformly in $\{1, \dots, n\}$. In both copies of A we seek for the optimal coin, only that in **min** we have that if $P(X = 1) = 1/2 - \epsilon$ then all coins are identical and in **max** all coins are identical if $P(X = 1) = 1/2 + \epsilon$. Each time **min** or **max** queries X we sample it, and when it queries a different random variable C_k we simulate such a coin. We define our output as follows. We consider only the copy that terminates first. When the **min** (**max**) copy returns i (j), which is the index of X , we claim that X has bias $1/2 + \epsilon$ ($1/2 - \epsilon$) and if it returns $k \neq i$ ($k \neq j$) we claim that X has bias $1/2 - \epsilon$ ($1/2 + \epsilon$).

For the correctness, with probability $1 - 2\delta$ both copies of A return the best arm. From now on we assume that each copy that terminates returns the best arm. If X has bias $1/2 - \epsilon$ ($1/2 + \epsilon$) then the **min** (**max**) copy has n i.i.d

random variables with bias $1/2 - \epsilon$ ($1/2 + \epsilon$). In such a case X and the other random variables are indistinguishable, and the chance X is selected as output is $1/n$ (since its index is chosen initially uniformly at random). Therefore, if $P(X = 1) = 1/2 - \epsilon$ and \min terminates first, with probability $1 - 1/n$ another arm is chosen and we have a correct output. But, with probability $1/n$ we have an error, since i is chosen. If $P(X = 1) = 1/2 - \epsilon$ and \max terminates first, since A does not err, its output is different from X , and we do not err. (The case when the bias is $1/2 + \epsilon$ is similar.) This implies that our algorithm is $(\epsilon, 2\delta + 1/n)$ -PAC. It remains to study our sample complexity.

We now analyze the sample complexity. We assume that the number of samples of X in \min and \max cannot differ in more than one sample. (Formally, there is a scheduler which schedules the copy that sampled X the least number of times. If they both sampled the same number of times, it chooses an arbitrary one.) Clearly, one of the copies terminates eventually. Since in one of the copies we have n identical random variables, its sample complexity is T/n . This implies that the expected sample complexity is $O(T/n)$. \square

Theorem 5. *Any (ϵ, δ) -PAC multi-armed bandit algorithm has an arm sample complexity of at least $T(\epsilon, \delta, n)$, where*

$$T(\epsilon, \delta, n) = \Omega\left(\frac{n \log(1/\delta')}{\epsilon^2}\right),$$

where $\delta' = 2\delta + 1/n$.

Proof. By Lemma 3 we have an (ϵ, δ') -PAC algorithm with sample complexity $O(T(\epsilon, \delta, n)/n)$, where $\delta' = 2\delta + 1/n$. The result follows from the $\Omega(\log(1/\delta')/\epsilon^2)$ lower bound on the expected sample complexity of the biases problem [8]. \square

Remark 3. For $\delta < 1/n$, we can give two simple lower bounds. The first is due to [1], which gives us $\Omega(\frac{\log(1/\delta)}{\epsilon^2})$. The second is due to a simple experiment which is composed from $n - 1$ arms with zero expectation and one with ϵ expectation. For this experiment any algorithm should sample $\Omega(\frac{n \log(1/\delta)}{\epsilon})$ times.

5 Markov Decision Processes

In this section we use the techniques developed for multi-armed bandit problems for deriving learning algorithms for MDPs. We show how to use any (ϵ, δ) -PAC multi-armed bandit algorithm to derive an algorithm for learning in an MDP. The learning algorithm in the MDP is based on Phased Q-learning [15]. Using our transformation we are even able to slightly improve the complexity of the Phased Q-Learning, mainly reduce its dependency on the number of action from $O(|A| \log |A|)$ to $O(|A|)$.

5.1 Phased Q-Learning

Phased Q-Learning [15] operates in phases, where in each phase every state action pair is sampled m times, and based on the new sample a new estimate to the Q function is derived.

Algorithm : **Phased Q-Learning**(n, m)

1. For every state-action pair (s, a) : $\hat{Q}_0(s, a) = 0$
2. For $k = 1$ to n DO
 - (a) For every state action pair invoke $sample(s, a)$ for m times, and let (s_i^k, r_i^k) be the i -th reply.
 - (b) Update:

$$\hat{Q}_{k+1}(s, a) = \frac{1}{m} \left(\sum_{i=1}^m r_i^k(s, a) + \gamma \hat{V}_k(s_i^k) \right),$$

- (c) $\hat{V}_{k+1}(s) = \max_a \hat{Q}_{k+1}(s, a)$

Theorem 6 ([15]). For $n = \frac{\ln(\frac{V_{max}}{2(1-\gamma)\epsilon})}{1-\gamma}$ and $m = O\left(\frac{\ln(|S||A|n/\delta)}{\epsilon^2(1-\gamma)^2}\right)$ the Phased Q-learning(n, m) converges with probability $1 - \delta$ to an ϵ -optimal policy.

As a consequence from the theorem we have that the sample complexity is:

$$nm|S||A| = O\left(\frac{|S||A|V_{max}^2}{\epsilon^2(1-\gamma)^3} \left(\ln(|S||A|/\delta) + \ln\left(\frac{\ln(V_{max}/\epsilon)}{1-\gamma}\right) \right) \ln\left(\frac{V_{max}}{(1-\gamma)\epsilon}\right)\right)$$

5.2 MAB Phased Q-learning algorithm

We like to first establish a simple connection between the multi-armed bandit problem and the Phased Q-learning algorithm. We do this by considering, for each phase and state, the update of the Phased Q-learning as a multi-armed bandit problem.

Note that during phase k of Phased Q-Learning, the value of $V_k(s)$ is fixed. 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 $P_{s, s'}^a$.

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 multi-armed bandit problem in order to find the best arm. In the following we show how, using a more sophisticated multi-armed bandit algorithm, 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 to learn an MDP. In order to use the multi-armed bandit algorithm B as, a black box, we define a simple interface, using the following procedures:

- $Init_B(\epsilon, \delta)$ - 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 $|r^* - v| \leq \epsilon$.)

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

Algorithm : **MAB Phased Q-Learning** (ϵ, δ)

1. Let $\hat{\epsilon} = \frac{\epsilon(1-\gamma)}{2}$; $n = \log_{\gamma}(\hat{\epsilon}/2V_{max}) = O(\ln(\frac{V_{max}}{(1-\gamma)\epsilon})/(1-\gamma))$;
 $\hat{\delta} = \frac{\delta}{|S|^n}$.
2. Initialize for every $s \in S$: $V_0(s) = 0$,
3. For $i = 1$ to n and for every $s \in S$ Do
 - (a) $Init_B(\hat{\epsilon}, \hat{\delta})$
 - (b) $a = GetArm_B()$
 - (c) $(s', r) = sample(s, a)$
 - (d) $r' = r + \gamma V_i(s')$
 - (e) $Update_B(a, r')$
 - (f) IF $Stop(a, v) = \text{FALSE}$ Then Go to (3b)
 ELSE $V_{i+1}(s) = v$; $\pi(s) = a$.
4. output π .

5.3 MAB Phased Q-Learning analysis

Let B be an (ϵ, δ) -PAC multi-armed bandit algorithm, and assume B has arm sample complexity $T_B(\epsilon, \delta)$. Namely, with probability $1 - \delta$, algorithm B terminates after at most $T_B(\epsilon, \delta)$ and outputs a policy π which is ϵ -optimal. In this subsection we analyze the MAB Phased Q-Learning algorithm, and derive its running time as a function of T_B .

Theorem 7. *Assume B is an $(\hat{\epsilon}, \hat{\delta})$ -PAC multi-armed bandit algorithm. MAB Phased Q-Learning (ϵ, δ) algorithm, with probability at least $1 - \delta$, outputs a policy π which is an ϵ -optimal policy, and has sample complexity*

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

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

Lemma 4. *Assume B is an $(\hat{\epsilon}, \hat{\delta})$ -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 $k \leq n$, $\|V^* - V_k\|$ is bounded by $\frac{\hat{\epsilon}}{1-\gamma} + V_{max}\gamma^k$.*

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

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

initializations of algorithm B and for each invocation the failure probability is bounded by $\frac{\delta}{|S|^n}$. 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,

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

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

We show by induction on the number of phases, that $d_k = \|V_k - \hat{V}_k\|_\infty \leq \frac{\hat{\epsilon}}{1-\gamma}$. 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.

$$\begin{aligned} |V_k(s) - \hat{V}_k(s)| &= \left| \max_u \left[\frac{1}{m_{s,u}} \sum_{i=1}^{m_{s,u}} r(s, u) + \gamma V_{k-1}(s'_i) \right] \right. \\ &\quad \left. - \max_a \left[E[R(s, a)] + \gamma \sum_{s'} P_{s,s'}^a \hat{V}_{k-1}(s') \right] \right| \\ &\leq \max_{\rho \in \{-\hat{\epsilon}, \hat{\epsilon}\}} \left| \max_u \left[E[R(s, u)] + \gamma \sum_{s'} P_{s,s'}^u V_{k-1}(s') \right] + \rho \right. \\ &\quad \left. - \max_a \left[E[R(s, a)] + \gamma \sum_{s'} P_{s,s'}^a \hat{V}_{k-1}(s') \right] \right| \\ &\leq \hat{\epsilon} + \max_a \left| \gamma \sum_{s'} P_{s,s'}^a (V_{k-1}(s') - \hat{V}_{k-1}(s')) \right| \\ &\leq \hat{\epsilon} + \gamma d_{k-1} \\ &\leq \hat{\epsilon} + \gamma \left(\frac{\hat{\epsilon}}{1-\gamma} \right) = \frac{\hat{\epsilon}}{1-\gamma}. \end{aligned}$$

To conclude the proof note that for the value iteration we have that $\|\hat{V}_k - V^*\| \leq \gamma^k V_{max}$, where $\hat{V}_0 = 0$ (see, e.g. [5]). \square

Lemma 5. *When the MAB Phased Q-Learning algorithm terminates, with probability at least $1 - \delta$, policy π is an ϵ -optimal policy.*

Proof. By Lemma 4 we have that with probability at least $1 - \delta$ the difference $\|V_k - V^*\| \leq \frac{\hat{\epsilon}}{1-\gamma} + V_{max} \gamma^k$. Since $\hat{\epsilon} = \frac{\epsilon(1-\gamma)}{2}$, we have that $\|V_k - V^*\| \leq \frac{\epsilon}{2} + V_{max} \gamma^k$. The lemma follows from our choice of $n = \log_\gamma \left(\frac{\hat{\epsilon}}{2V_{max}} \right)$. \square

Now we can prove Theorem 7.

Proof. The correctness follows from Lemma 5. We bound the sample complexity as follows. By definition, the MAB Phased Q-Learning algorithm samples at each state and action during every phase $T_B(\hat{\epsilon}, \hat{\delta})$. By definition of the algorithm, the

number of phases is $n = O\left(\frac{\ln(V_{max}/\epsilon)}{1-\gamma}\right)$, and each phase is composed from $|S|$ MAB instances. This completes the bound on the sample complexity. \square

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

$$O\left(\frac{|S| |A| V_{max}^2}{(1-\gamma)^3 \epsilon^2} \ln\left(\frac{V_{max}}{(1-\gamma)\epsilon}\right) \ln\left(\frac{|S| \ln(V_{max}/\epsilon)}{\delta(1-\gamma)}\right)\right).$$

Remark 4. One can use the successive elimination algorithm as the basic MAB algorithm. This would result in improved convergence rates if the Q functions at the end of each phase are well separated. We note that by using the successive elimination algorithm with finite stopping time (as suggested in Remark 2) the convergence rate is as good as the rates of [15]. If, however, one is “lucky” and during the process of the phased Q-learning the Q values are well separated in the various rounds, then the convergence rates may improve significantly. It is an open question if properties of the model itself (such as the separation of the optimal Q function) may lead to improved convergence rates.

Acknowledgements. This research was supported in part by a grant from the Israel Science Foundation. S.M. would like to thank Nahum Shimkin for helpful discussions.

References

1. M. Anthony and P.L. Bartlett. *Neural Network Learning; Theoretical Foundations*. Cambridge University Press, 1999.
2. 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.
3. P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. preprint, 2001.
4. D.A. Berry and B. Fristedt. *Bandit Problems*. Chapman and Hall, 1985.
5. D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1995.
6. V.S. Borkar and S.P. Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control.*, 38(2):447–469, 2000.
7. R. Brafman and M. Tennenholtz. R-MAX – A General Polynomial Time Algorithm for Near Optimal Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*, 2001.
8. H. Chernoff. *Sequential Analysis and Optimal Design*. Society for industrial and Applied Mathematics, Philadelphia, 1972.

9. P. Dayan and C. Watkins. Q-learning. *Machine Learning*, 8:279–292, 1992.
10. Eyal Even-Dar and Yishay Mansour. Learning rates for q-learning. In *Fourteenth Annual Conference on Computational Learning Theory*, pages 589–604, 2001.
11. C. N. Fiechter. PAC adaptive control of linear systems. In *Tenth Annual conference on Computational Learning Theory*, pages 72–80, 1997.
12. J. Gittins and D. Jones. A dynamic allocation index for the sequential design of experiments. In J. Gani, K. Sarkadi, and I. Vincze, editors, *Progress in Statistics*, pages 241–266. North-Holland, Amsterdam, 1974.
13. M. Kearns, Y. Mansour, and A. Ng. Approximate planning in large POMDPs via reusable trajectories. In *Advances in Neural Information Processing Systems*, 1999.
14. M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *Proc. of the 15th Int. Conf. on Machine Learning*, pages 260–268. Morgan Kaufmann, 1998.
15. M. Kearns and S. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms near-optimal reinforcement learning in polynomial time. In *Neural Information Processing Systems 11*, pages 996–1002. Morgan Kaufmann, 1999.
16. Michael J. Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov Decision Processes. In *International Joint Conference on AI*, pages 1324–1231, 1999.
17. T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.
18. H. Robbins. Some aspects of sequential design of experiments. *Bull. Amer. Math. Soc.*, 55:527–535, 1952.
19. J.N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.