Approximately Optimal Approximate Reinforcement Learning

Sham Kakade SHAM@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL, London WC1N3AR, UK

John Langford JCL@CS.CMU.EDU
Computer Science Department, Carnegie-Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15217

Abstract

We make a reduction of the reinforcement learning problem to solving two types of problems — "How do we gain access to states?" (exploration) and "How do we solve a regression problem?" (supervised learning). Given solutions to each of these problems, our new algorithm: (1) is guaranteed to improve a performance metric (2) is guaranteed to terminate in a 'small' number of timesteps and (3) returns an "approximately" optimal policy. The quantified statements of (2) and (3) depend on the quality of the supervised learning problem solution, but *not* explicitly on the the size of the state space.

1 Introduction

In order to solve realistic reinforcement learning problems, it is critical that approximating algorithms be used. Unfortunately, the two standard approaches, greedy dynamic programming and policy gradient methods, can fail to efficiently improve the policy. Approximate value function methods suffer from a lack of theoretical performance guarantees. We show how policy gradient techniques can require an unreasonably large number of samples in order to determine the gradient accurately. This is due to policy gradient method's fundamental intertwining of "exploration" and "exploitation".

Drawing upon the strengths of each approach we propose a new reinforcement learning algorithm based on two critical observations. First, we need to improve the policy at states that are visited infrequently under the current policy, since policy improvement at these states could ultimately lead to large policy improvement, *ie* we need to incorporate *exploration*. The second is that we must quantify how much policy improvement is possible with an approximate scheme. At one extreme, greedy methods show that if a large policy update is made then the policy could significantly degrade, while at the other extreme, for a sufficiently small change to the policy, gradient methods guarantee performance improvement. We connect these two extremes, to determine how much policy improvement is possible at finite step sizes. This allows us to determine how long it takes to obtain some 'approximately' optimal policy. We then show the quality of this 'approximately' policy is determined both by the level of approximation used and the how 'exploratory' the start-state distribution is.

The algorithm we present overcomes the problems with both gradient methods and greedy dynamic programming.

2 Preliminaries

A reinforcement learning problem is a finite Markov decision process (MDP) with rewards. It is defined by the tuple $(S, D, A, \mathcal{R}, P)$ where: S is finite set of states, D is the starting state distribution, A is a finite set of actions, $\mathcal{R}$ is a reward function $\mathcal{R} : S \times A \to [0, R]$, and P is the transition model in which p^a_{ij} is the transition probability from state i to j under action a.

The agent's decision making procedure is characterized by a stochastic policy $\pi(a|s)$, which is the probability of taking action a in state s. Although in many situations, the goal of the agent is to maximize the average reward, we only consider the case of maximizing the γ-discounted average reward from the starting state distribution D, which has a similar solution for processes that 'mix' on a reasonable timescale [4]. It is convenient to define a distribution over future states, with respect to some arbitrary start-state distribution μ, in the following way:

$$d(s|\pi, \mu, \gamma) \equiv (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s|\pi, \mu) \quad (2.1)$$

where $\Pr(s_t = s|\pi, \mu)$ is the probability that the state at time t is s, under the policy π and the start-state distribution μ. This is analogous to the stationary distribution in the infinite time setting, since as $\gamma \to 1$, $d(s|\pi, \mu, \gamma)$ tends to the stationary distribution. For brevity, we will sometimes write this as $d(s|\pi)$ when γ and μ are clear from context.

Let $\delta(s)$ be the distribution over states which concentrates all mass on state s. We define the value function as the expected reward under $\pi(a|s)d(s|\pi, \delta(s))$, or

$$J_\pi(s) \equiv E_{a,s' \sim \pi(a|s')d(s'|\pi,\delta(s),\gamma)} \mathcal{R}(s', a) \tag{2.2}$$

which is equivalent to the more-standard definition $J_\pi(s) \equiv (1-\gamma)E\{\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)|\pi, s_0 = s\}$. Note that we are using *normalized* values. The state-action value is defined as: $Q_\pi(s, a) \equiv (1-\gamma)\mathcal{R}(s, a) + \gamma \sum_{s'} p^a_{ss'} J_\pi(s')$, and the advantage is defined as $A_\pi(s, a) \equiv Q_\pi(s, a) - J_\pi(s)$. We will sometimes suppress the π subscript when the policy is clear from context.

The goal of the agent is to maximize the discounted reward from the start state distribution D,

$$\eta(\pi) \equiv E_{s \sim D}\{J_\pi(s)\}.$$

A well known result is that a solution exists which simultaneously maximizes $J_\pi(s)$ for all states.

3 The Problems with Current Methods

We now examine in more detail the problems with approximate value function methods and policy gradient methods. There are three questions to which we desire answers to:

(1) Is there some performance measure that is guaranteed to improve at every step?
(2) How difficult is it to verify if a particular update will improve this measure?
(3) After a reasonable number of policy updates, what performance level is obtained?

We now argue that both greedy dynamic programming and gradient methods give unsatisfactory answers to these questions. Note that one question we did not ask is "What is the quality of the asymptotic policy?". We are only interested in policies that we can find in a reasonable amount of time. Understanding the problems with these current methods will give insight into our new algorithm, which addresses these three questions.

3.1 The Weakness of Approximate Value Function Methods

Approximate value function methods suffer from a paucity of theoretical results on the performance of a policy based on the approximate values. This leads to weak answers to all three questions.

Let us consider some function approximator $f(s)$ with the l_∞-error:

$$\varepsilon = \max_s |f(s) - J_\pi(s)|$$

where π is some policy. Let J_g the value of a greedy policy based on this approximation. We have the following guarantee (see [2]) for all states s:

$$J_g(s) \geq J_\pi(s) - \frac{2\gamma\varepsilon}{1-\gamma}. \tag{3.1}$$

In other words, the performance is guaranteed to not decrease by more than $\frac{2\gamma\varepsilon}{1-\gamma}$. Question 2 is not applicable since these methods don't guarantee improvement and a performance measure to check isn't well defined.

For approximate methods, the time required to obtain some performance level is not well understood. At best, some asymptotic results exist (see [3]) . In contrast, exact value function methods have bounds showing how fast the values converge to optimal (see [2]).

3.2 The Convergence Problem for Policy Gradients

For policy gradient techniques, question 1 has the appealing answer that the performance measure of interest is guaranteed to improve under gradient descent. We now address question 2 by examining the situations in which estimating the gradient direction will be difficult. We show that the lack of exploration in gradient methods translates into requiring a very large number of timesteps in order to accurately estimate the gradient *direction*.

Consider the simple MDP shown in Figure 3.1 (adapted from [8]). Under a policy that gives equal probability to all actions, the expected time to the goal from the left most state is $3(2^n - n - 1)$, and with $n = 50$, the expected time to the goal is about 10^{15}. In general, many MDPs behave in the following manner: random actions are more likely than not to increase the distance to the goal state. For these classes of problems (see [9]), the expected time to reach the goal state using undirected exploration, *ie* random walk exploration, is exponential in the size of the state space. Thus, any on-policy method will have to run for at least this long before any policy improvement can occur. In online value function methods, this problem is seen as a lack of exploration.

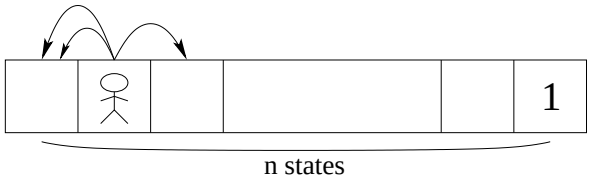

Figure 3.1. Example 1: Two actions move agent to the left and one actions moves agent to the right.

Any sensible estimate of the gradient without reaching the goal state would be zero, and obtaining non-zero estimates requires exponential time with on-policy samples. Importance sampling methods do exist (see [5, 6]), but are not feasible solutions. The reason is that if the agent could follow some 'off-policy' trajectory to reach the goal state in a reasonable amount of time, the importance weights would have to be exponentially large.

Note that a zero estimate is a rather accurate estimate of the gradient in terms of magnitude, but this provides no information about direction. The direction of the gradient is the crucial quantity of interest since we must step in the "right" direction in order to improve our policy. Some analysis has been done studying the deviation of gradient estimates [1]. However, this analysis focused on obtaining the policy gradient accurately (within some ε tolerance) rather than on obtaining the correct sign. This is a crucial difference because the magnitude of the gradient can be very small when the policy is far from optimal.

Let us give an additional example demonstrating the problems for a simple two state MDP as shown in Figure 3.2A, which uses the common Gibbs table-lookup distribution, $\tilde{\Pi}_G = \{\pi_\theta : \pi(a|s) \propto \exp(\theta_{sa})\}$. Increasing the chance of a self-loop at i decreases the stationary probability of j, which hinders the learning at state j. Under an initial policy that has the stationary distribution $\rho(i) = .8$ and $\rho(j) = .2$ (using $\pi(a_1|i) = .8$ and $\pi(a_1|j) = .9$), learning at state i reduces the learning at state j leading to an an extremely flat plateau of improvement at 1 average reward shown in Figure 3.2B. Figure 3.2C shows that this problem is so severe that $\rho(j)$ drops as low as 10^{-7} from it's initial probability of .2. As in example 1, to obtain a nonzero estimate of the gradient it is necessary to visit state j. The situation could be even worse with a few extra states in a chain as in figure 3.1.

Although in the limit a good policy might be found, these results do not bode well for the answer to question 3, which is concerned with how fast we can get there. These results suggest that in any reasonable number of steps, a gradient method could end up being

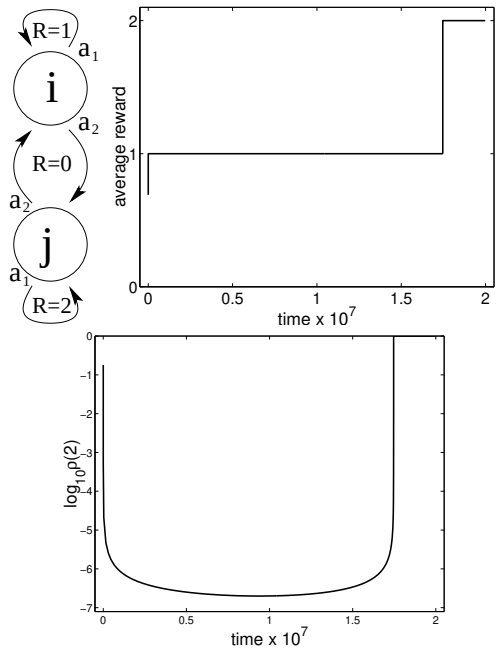

Figure 3.2. Example 2: A) See text. B) The average reward vs. time (on a 10^7 scale) of a policy under standard gradient descent in the limit of an infinitesimally small learning rate (initial conditions stated in text). C) The stationary probability of state j in B vs. time (on a 10^7 scale).

trapped at plateaus where estimating the gradient direction has an unreasonably large sample complexity. The asymptotic results of standard stochastic gradient ascent techniques show that as the number of samples go to ∞ a local optima is found. Unfortunately, these asymptotic results hide the sample complexity issues outlined here. Answering question 3 is crucial to understand how well gradient methods perform, and (to our knowledge) no such analysis exists.

4 Approximately Optimal RL

The fundamental problem with policy gradients is that η, which is what we ultimately seek to optimize, is insensitive to policy improvement at unlikely states though policy improvement at these unlikely states might be necessary in order for the agent to achieve near optimal payoff. Note that in example 1, policy improvement at state 49 will have little impact on the current performance, since the agent hardly ever reaches state 49 anyway. However, policy improvement at these unlikely states can ultimately lead to drastic policy improvement.

We desire an alternative performance measure that does not down weight advantages at unlikely states

or unlikely actions. A natural candidate for a performance measure is to weight the improvement from all states more uniformly (rather than by D), such as

$$\omega(\pi) = E_{s\sim\mu}\{J_\pi(s)\}\,,$$

where μ is some 'exploratory' starting state distribution. To this end, ***we assume we have the ability to use an arbitrary start-state distribution*** μ. Note that for the special case when $\mu = D$ then $\omega = \eta$.

Any optimal policy simultaneously maximizes both ω and η. Unfortunately, the policy that maximizes ω within some restricted class of policies may have poor performance according to η. In general, the quality of the optima of ω is related to the the quality of the optima under η.

Greedy policy iteration updates the policy to some π' based on some approximate state-action values. Instead, let us consider the following more conservative update rule:

$$\pi_{\text{new}}(a|s) = (1-\alpha)\pi(a|s) + \alpha\pi'(a|s)\,, \tag{4.1}$$

for some π' and $\alpha \in [0,1]$. To guarantee improvement with $\alpha = 1$, π' must choose a better action at every state, or else we could suffer the penalty shown in equation 3.1.

4.1 Policy Improvement

A more reasonable situation is one in which we are able to improve the policy with some $\alpha > 0$ using a π' that chooses better actions at most but not all states. Let us define the *policy advantage* $\mathbb{A}_{\pi,\mu}(\pi')$ of some policy π' with respect to the policy π and the start-state distribution μ to be:

$$\mathbb{A}_{\pi,\mu}(\pi') \equiv E_{s\sim d(s|\pi,\mu)}\left[\sum_a \pi'(a|s)A_\pi(s,a)\right]\,.$$

Intuitively, this measures the expected advantage of π' over π, with respect to the set of states visited under π. Note that a policy found by one step of policy improvement will maximize the policy advantage.

The gradient of a parameterized policy is

$$\nabla\omega = \sum_{s,a} d(s|\pi,\mu)\nabla\pi(a|s)Q_\pi(s,a)\,,$$

(see [7]). It is straightforward to show that $\frac{\partial\omega}{\partial\alpha}|_{\alpha=0} = \frac{1}{1-\gamma}\mathbb{A}_{\pi,\mu}$, so the change in ω is:

$$\Delta\omega = \frac{\alpha}{1-\gamma}\mathbb{A}_{\pi,\mu}(\pi') + O(\alpha^2)\,. \tag{4.2}$$

Hence, for sufficiently small α, policy improvement can occur if the policy advantage is positive, and at the other extreme of $\alpha = 1$, significant degradation could occur. We now connect these two regimes to determine how much policy improvement is possible.

Theorem 4.1. *Let $\mathbb{A}$ be the policy advantage of π' with respect to π and μ. and let $\varepsilon = \max_s|\sum_a \pi'(a|s)A_\pi(s,a)|$. For the update rule 4.1 and for all $\alpha \in [0,1]$:*

$$\omega(\pi_{new}) - \omega(\pi) \geq \frac{\alpha}{1-\gamma}(\mathbb{A} - \frac{2\alpha\gamma\varepsilon}{1-\gamma(1-\alpha)})\,.$$

It is possible to construct a two state example showing this bound is tight for all α though we do not provide this example here.

The quantity ε is the average advantage, under π', with the largest magnitude. The first term is analogous to the first order increase specified in equation 4.2, and the second term is a penalty term. Note that if $\alpha = 1$, the bound reduces to

$$\omega(\pi_{\text{new}}) - \omega(\pi) \geq \frac{\mathbb{A}}{1-\gamma} - \frac{2\gamma\varepsilon}{1-\gamma}$$

and the penalty term is identical to that of greedy dynamic programming, where the ε defined here is analogous to max-norm error (see equation 3.1). Thus the bound agrees with the local information provided by the gradient and greedy penalty when $\alpha = 1$.

The following corollary shows that the greater the policy advantage the greater the guaranteed increase in performance.

Corollary 4.2. *Let R be the maximal possible reward and $\mathbb{A}$ be the policy advantage of π' with respect to π and μ. If $\mathbb{A} \geq 0$, then using $\alpha = \frac{(1-\gamma)\mathbb{A}}{4R}$ will guarantee the following policy improvement:*

$$\omega(\pi_{new}) - \omega(\pi) \geq \frac{\mathbb{A}^2}{8R}\,.$$

Proof. Using the previous theorem, it is straightforward to show the change is bounded by $\frac{\alpha}{1-\gamma}(\mathbb{A} - \alpha\frac{2R}{1-\gamma})$. The corollary follows by choosing the α that maximizes this bound. □

4.2 The answer to question 3

We address question 3 by first addressing how fast we converge to some policy then bounding the quality of this policy. Naively, we expect our ability to obtain policies with large advantages to affect the speed of improvement and the quality of the final policy. Instead of explicitly suggesting algorithms that obtain policies with large advantages, we assume access to an algorithm that solves this problem. Let us call this ε-good algorithm $BlackBox_\varepsilon(\pi,\mu)$, which is defined as:

Definition 4.3. $BlackBox_\varepsilon(\pi,\mu)$ is a function of a policy π and start-state distributions μ which returns a policy π' such that $\mathbb{A}_{\pi,\mu}(\pi') \geq \text{OPT}(\mathbb{A}_{\pi,\mu}) - \varepsilon$, where $\text{OPT}(\mathbb{A}_{\pi,\mu}) = \max_{\pi'}\mathbb{A}_{\pi,\mu}(\pi')$.

We drop the μ dependence when clear. In the discussion, we show how a supervised learning algorithm that fits the advantages with an l_1-error of $\frac{\varepsilon}{2}$ is sufficient to construct such a $BlackBox_\varepsilon$ (though direct policy methods could also be sufficient).

The "breaking point" of our supervised learning algorithm is reached when it is no longer guaranteed to return a policy with a positive policy advantage (i.e. when $\mathrm{OPT}(\mathbb{A}_{\pi,\mu}) - \varepsilon < 0$).

For simplicity, we assume we have knowledge of ε. A crude outline of the *Conservative Policy Iteration* algorithm is:

(1) Call $BlackBox_\varepsilon(\pi, \mu)$ to obtain some π'
(2) Use $O(\frac{R^2}{\varepsilon^2} \log \frac{R^2}{\delta\varepsilon^2})$ samples to obtain a Monte Carlo estimate $\hat{\mathbb{A}}$ of $\mathbb{A}_{\pi,\mu}(\pi')$.
(3) If $\hat{\mathbb{A}} < \frac{2\varepsilon}{3}$, STOP and return π.
(4) If $\hat{\mathbb{A}} \geq \frac{2\varepsilon}{3}$, then update policy π according to equation 4.1 using $\frac{(1-\gamma)(\hat{\mathbb{A}} - \frac{\varepsilon}{3})}{4R}$ and return to step 1.

The following theorem shows that in polynomial time, the algorithm finds a policy π that is 'close' to the break-point of the black box.

Theorem 4.4. *With probability at least $1-\delta$, conservative policy iteration will: i) improve ω with every policy update, ii) cease in at most $72\frac{R^2}{\varepsilon^2}$ calls to $BlackBox_\varepsilon(\pi, \mu)$, and iii) return a policy π such that $OPT(\mathbb{A}_{\pi,\mu}) < 2\varepsilon$.*

The proof is in the next section.

To complete the answer to question 3, we need to address the quality of the policy found by this algorithm. Note that our algorithm ceases in an amount of time that does not depend on the start-state distribution μ used though the policy π that we find will depend on μ since $\mathrm{OPT}(\mathbb{A}_{\pi,\mu}) < 2\varepsilon$. If $d(s|\mu)$ is highly non-uniform, then a small policy advantage will not necessarily imply that all advantages will be small. If there are any large advantages, the policy could be far from optimal. The following theorem quantifies this. We use standard definition of the l_∞-norm for some vector $f(s)$, $||f||_\infty \equiv \max_s f(s)$.

Theorem 4.5. *(simple version) Assume we have some π such that $OPT(\mathbb{A}_\pi) < \varepsilon$. Let π^* be an optimal policy. Then*

$$\max_s |J_\pi(s) - J_{\pi_*}(s)| \leq \frac{\varepsilon}{(1-\gamma)} \left\| \frac{d(s|\pi^*, \mu')}{d(s|\pi, \mu)} \right\|_\infty .$$

The proof is provided later in the paper. This bound elucidates the problem of using the given start-state distribution, D, instead of a more uniform distribution. The reason is that if $d(s|\pi, D)$ does not give weight to any state where the optimal visits, then the policy could be far from optimal. The factor of $\left\| \frac{d(s|\pi^*, \mu')}{d(s|\pi, \mu)} \right\|_\infty$ represents a mis-match between the distribution of states of the current policy and that of the optimal policy. Intuitively, weight must be given to all states that the optimal could visit. The use of a uniform metric will bound $d(s|\pi, \text{uniform}) \geq \frac{1-\gamma}{\text{size of state space}}$. Essentially, exploration is necessary to improve the policy at these unlikely states.

We now discuss and prove these theorems in more detail.

5 Conservative Policy Iteration

After our black box returns some policy π', we must address how to update the policy. If we knew the policy advantage of this policy was $\mathbb{A}$ and if $\mathbb{A} > 0$, then we could use $\alpha = \frac{(1-\gamma)\mathbb{A}}{4R}$.

Let us address how many samples are required to accurately determine $\mathbb{A}_\pi(\pi')$. We must specify a sampling model to estimate the policy advantage. It is straightforward to show that the policy advantage can be written as $\mathbb{A}_{\pi,\mu}(\pi') = E_{s\sim d(s|\pi,\mu)}\left[\sum_a (\pi'(a|s) - \pi(a|s))Q(s,a)\right]$ which is a more convenient form since we can directly estimate the state-action values.

Each sample required will consist of a sample s from from $d(s|\pi, \mu)$, an a from the uniform distribution, and a nearly unbiased estimate $\hat{Q}(s,a)$ of $Q_\pi(s,a)$. Then, we can use Monte Carlo integration in order to estimate the value of $\mathbb{A}_{\pi,\mu}(\pi')$. Given certain assumptions, it is reasonable to suppose that we can obtain these samples in an on-policy manner, using a sample from μ and $O(\frac{1}{1-\gamma})$ transitions in our MDP. This can be done by first drawing a state s from $d(s|\pi, \mu)$, which can be done by acting in the MDP. Then we choose an action a from the uniform distribution and obtain an estimate $\hat{Q}(s,a)$ of the $Q(s,a)$ value, by acting according to the policy for $O(\frac{1}{1-\gamma})$ steps. We use importance sampling with a uniform proposal distribution over the actions to estimate the policy advantage. We do not use π or π' as the importance distribution, since if either of these are deterministic the importance weights may be large. The estimate of the policy advantage from the i-th sample will be $x_i = n_a \hat{Q}_i(s,a)(\pi'(a|s) - \pi(a|s))$ where n_a is the number of actions.

How many samples are required? Since $\hat{Q}_i \in [0, R]$, our samples satisfy $x_i \in [-n_a R, n_a R]$. Using Hoeffding's inequality for k independent identical random

variables, we have:

$$\Pr\left(|\mathbb{A} - \hat{\mathbb{A}}| > \delta\right) \leq 2e^{-\frac{k\delta^2}{2n_a^2 R^2}}, \tag{5.1}$$

where $\hat{\mathbb{A}} = \frac{1}{k}\sum_{i=1}^{k} x_i$. Hence, the number of samples required to obtain a δ accurate sample with a fixed error rate is $O\left(\frac{n_a^2 R^2}{\delta^2}\right)$. Consequently, if the policy advantage is large then few samples will be needed.

Now the proof of the conservative policy iteration algorithm is straightforward:

Proof. There are three statements, that must all hold with probability $1-\delta$, — ω improves at every step, the algorithms ceases in $\frac{72R^2}{\varepsilon^2}$ iterations, and the policy found has maximal policy advantage less than 2ε. Let us assume that every time we make a Monte-Carlo estimate of the policy advantage, our estimate $\hat{\mathbb{A}}$ of the policy advantage satisfies $|\mathbb{A} - \hat{\mathbb{A}}| < \frac{\varepsilon}{3}$. We estimate the probability this assumption fails later.

The assumption implies that if we observe $\hat{\mathbb{A}} \geq \frac{2\varepsilon}{3}$, then $\mathbb{A} \geq \frac{\varepsilon}{3}$. By corollary 4.2, step 4 will guarantee improvement of $\frac{\left(\hat{\mathbb{A}}-\frac{\varepsilon}{3}\right)^2}{8R} \geq \frac{\varepsilon^2}{72R}$ using $\alpha = \frac{(1-\gamma)\left(\hat{\mathbb{A}}-\frac{\varepsilon}{3}\right)}{4R}$. Since $0 \leq \omega \leq R$, there are at most $\frac{72R^2}{\varepsilon^2}$ steps before ω becomes larger than R, so the algorithm must cease in this time. Our algorithm ceases and returns π when we observe that the policy advantage of some π' satisfies $\hat{\mathbb{A}}_\pi(\pi') < \frac{2\varepsilon}{3}$ which implies $\mathbb{A}_\pi(\pi') < \varepsilon$. By definition of the black box, it follows that $\mathrm{OPT}(\mathbb{A}_\pi) < 2\varepsilon$.

To obtain an error probability of less than δ, we must obtain enough samples such that $|\mathbb{A} - \hat{\mathbb{A}}| < \frac{\varepsilon}{3}$ for every loop of the algorithm, and there are at most $\frac{72R^2}{\varepsilon^2}$ loops. By the union bound and inequality 5.1, we have $\frac{72R^2}{\varepsilon^2} 2e^{-\frac{k\varepsilon^2}{2n_a^2 R^2}} < \delta$, and so when $O\left(\frac{R^2}{\varepsilon^2}\log\frac{R^2}{\delta\varepsilon^2}\right)$ samples are used, the probability of failure is less than δ. □

6 How Good is The Policy Found?

The speed with which our algorithm ceases does not depend on the start-state distribution used. However, we now show that the quality of the resulting policy could strongly depend on this distribution.

The following lemma will be useful:

Lemma 6.1. *For any policy $\tilde{\pi}$ and start-state distribution μ,*

$$\omega_{\tilde{\pi}}(s) - \omega_\pi(s) = \frac{1}{1-\gamma} E_{s,a\sim d(s|\tilde{\pi},\mu)\tilde{\pi}}\{A_\pi(s,a)\}$$

Proof. Let $d_t(s)$ be probability that the state at time t is s while following $\tilde{\pi}$ under the start-state distribution μ. We have

$$\begin{aligned}
&\omega(\tilde{\pi}) \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t E_{s,a\sim d_t\tilde{\pi}}\{\mathcal{R}(s,a)\} \\
&= \sum_{t=0}^{\infty}\gamma^t E_{s,a\sim d_t\tilde{\pi}}\{(1-\gamma)\mathcal{R}(s,a) - J_\pi(s) + J_\pi(s)\} \\
&= E_{s\sim d_0}\{J_\pi(s)\} \\
&\quad + \sum_{t=0}^{\infty}\gamma^t E_{s',s,a\sim d_{t+1}d_t\tilde{\pi}}\{(1-\gamma)\mathcal{R}(s,a) + \gamma J_\pi(s') - J_\pi(s)\} \\
&= \omega(\pi) + \sum_{t=0}^{\infty}\gamma^t E_{s,a\sim d_t\tilde{\pi}}\{A_\pi(s,a)\} \\
&= \omega(\pi) + \frac{1}{1-\gamma} E_{s,a\sim d(s|\tilde{\pi},\delta(s))\tilde{\pi}}\{A_\pi(s,a)\}.
\end{aligned}$$

□

This lemma elucidates a a fundamental *measure mismatch* in reinforcement learning, which we now analyse. The performance measure of interest, $\eta(\pi)$ (defined with respect to start-state distribution D), changes in proportion the policy advantage $\mathbb{A}_{\pi,D}(\pi')$ for small α when using our update rule (see equation 4.2). A resulting improvement of $\frac{\mathbb{A}^2}{8R}$ can be guaranteed if $\mathbb{A} \geq 0$. Recall that the policy advantage is just the expected advantage under the distribution $d(s|\pi, D)\pi'(a|s)$. However, the difference between η and η^* is the average advantage under $d(s|\pi^*, D)\pi^*(a|s)$, where π^* is an optimal policy. Thus, if $\mathbb{A}_{\pi,D}(\pi')$ is small and D is highly non-uniform, this will not imply that η is close to optimal. For example, there could still be some large advantages where $d(s|\pi, D)$ is small and $d(s|\pi^*, D)$ is large. Also, with $\mathbb{A}_{\pi,D}(\pi')$ small, little improvement is guaranteed and even detecting if the policy advantage is positive ($\mathbb{A}_{\pi,D}(\pi') \geq 0$) will have large sample complexity. Intuitively, this is the situation where large advantages are at states with low probabilities of being visited under the current policy (as in examples 1 and 2).

We now quantify how far from optimal the resulting policy is using some start-state distribution μ.

Theorem 6.2. *Assume we have some π such that $OPT(\mathbb{A}_{\pi,\mu}) < \varepsilon$. For any policy $\tilde{\pi}$ and for any μ',*

$$\begin{aligned}
\omega_{\mu'}(\tilde{\pi}) - \omega_{\mu'}(\pi) &\leq \frac{\varepsilon}{(1-\gamma)}\left\|\frac{d(s|\tilde{\pi},\mu')}{d(s|\pi,\mu)}\right\|_\infty \\
&\leq \frac{\varepsilon}{(1-\gamma)^2}\left\|\frac{d(s|\tilde{\pi},\mu')}{\mu}\right\|_\infty.
\end{aligned}$$

Proof. The optimal policy advantage is $\mathrm{OPT}(\mathbb{A}_{\pi,\mu}) = \sum_s d(s|\pi,\mu) \max_a A_\pi(s,a)$. Therefore,

$$\begin{aligned}
\varepsilon &> \sum_s \frac{d(s|\pi,\mu)}{d(s|\tilde{\pi},\mu')} d(s|\tilde{\pi},\mu') \max_a A_\pi(s,a) \\
&\geq \min_s \left(\frac{d(s|\pi,\mu)}{d(s|\tilde{\pi},\mu')} \right) \sum_s d(s|\tilde{\pi},\mu') \max_a A_\pi(s,a) \\
&\geq \left\| \frac{d(s|\tilde{\pi},\mu')}{d(s|\pi,\mu)} \right\|_\infty^{-1} \sum_{s,a} d(s|\tilde{\pi},\mu') \tilde{\pi}(a|s) A_\pi(s,a) \\
&= (1-\gamma) \left\| \frac{d(s|\tilde{\pi},\mu')}{d(s|\pi,\mu)} \right\|_\infty^{-1} (\omega_{\mu'}(\tilde{\pi}) - \omega_{\mu'}(\pi))
\end{aligned}$$

where the last step follows from lemma 6.1. The second inequality is due to $\min_s d(s|\pi,\mu) \leq (1-\gamma)\min_s \mu(s)$. □

Note that $\left\| \frac{d(s|\pi^*,\mu')}{\mu} \right\|_\infty$ is a measure of the mismatch in using μ rather than the start-state distribution of an optimal policy. The interpretation of each factor of $\frac{1}{1-\gamma}$ is important. In particular, one factor of $\frac{1}{1-\gamma}$ is due to the fact that policy is $\frac{1}{1-\gamma}$ times the average advantage (under $d(s|\pi^*)$) away from optimal and another factor of $\frac{1}{1-\gamma}$ is due to the inherent un-uniformity of $d(s|\pi,\mu)$.

7 Discussion

We have provided an algorithm that finds an "approximately" optimal solution that is polynomial in the approximation parameter ε, but not in the size of the state space. We discuss the difficulty of constructing our *BlackBox* function and also make a few other points.

7.1 The Black Box Algorithm

The ability to find a policy with a large policy advantage can be stated as a supervised learning problem, though we do not specify the sample complexity of this supervised learning problem. Let us consider the loss given by:

$$E_{s\sim d(s|\pi,\mu)} \max_a |A_\pi(s,a) - f_\pi(s,a)| .$$

This loss is an *average* loss over the state space (though it is an ∞-loss over actions). It is in general much more tractable to deal with an average loss than an ∞-loss over the state-space. In general, it is the ∞-loss that is the relevant measure for greedy dynamic programming (see equation 3.1 and [3]).

It is straightforward to see that if we can keep this error below ε, then we can construct an 2ε-good black box by choosing $\pi'(a|s)$ to be nonzero only for actions in $\mathrm{argmax}_a f(s,a)$. Thus, the difficulty of constructing an 2ε-good black box is at most as difficult as solving a regression problem with ε loss.

Direct policy search methods could also be used to construct this *BlackBox*. Equation 5.1 states that the sample complexity of determining the policy advantage to δ accuracy is $O\left(\frac{n_a^2 R^2}{\delta^2}\right)$, which is related to the sample complexity of searching a set of polices to obtain one with the largest policy advantage.

7.2 What about improving η?

Eventhough we ultimately seek to have good performance measure under η (which is under D), we show that it is important to improve the policy under a somewhat uniform measure. An interesting question is "Can we improve the policy according to both η and ω (which is under μ) at each update?" In general the answer is "no", but consider improving the performance under $\tilde{\mu} = (1-\beta)\mu + \beta D$ instead of just μ. This metric only slightly changes the quality of the asymptotic policy. However, by giving weight to D, the possibility of improving η is allowed if the optimal policy has large advantages under D. Essentially if $\mathrm{OPT}(\mathbb{A}_{\pi,D})$ is large (greater than $\frac{\varepsilon}{\beta}$) , then finding a policy with large policy advantage with respect to $\tilde{\mu}$, will also find a policy with large policy advantage under D. Joint improvement will be possible for this case, with an appropriate setting of α.

The only situation where joint improvement with η is not possible is when $\mathrm{OPT}(\mathbb{A}_{\pi,D})$ is small. However, this is just the usual problematic case where, under D, the large advantages are inaccessible.

7.3 Prior Knowledge

A very interesting suggestion of how to chose μ, is to use prior knowledge of which states an optimal policy tends to visit or where any good policy tends to visit. The bounds we have presented show directly the importance of ensuring the agent starts in states where the optimal tends to visit.

7.4 Practical Variants

The algorithms we have suggested leads to a variety of practical variants. The most pragmatic for doing greedy dynamic programming is to fit the value functions under a more uniform metric and to use a line search over α with the greedy policy, rather than using the greedy $\alpha = 1$. Further tricks must be done to ensure a compact representation of the policy. The most pragmatic variant for using gradient methods is to use

a more uniform start state distribution, and then decay the distribution μ to D.

Acknowledgments

We give many thanks to Peter Dayan for suggestions on what constitutes efficient learning.

References

[1] P. Bartlett and J. Baxter. Estimation and approximation bounds for gradient-based reinforcement learning. Technical report, Australian National University, 2000.

[2] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Volumes 1 and 2.* Athena Scientific, 1995.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[4] S. Kakade. Optimizing average reward using discounted rewards. *In Proc. of Computational Learning Theory*, 2001.

[5] L. Peshkin an K. Kim N. Meuleau. Exploration in gradient-based reinforcement learning. Technical report, Massachusetts Institute of Technology, 2001.

[6] D. Precup, R. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. *ICML, in press*, 2001.

[7] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems*, 13, 2000.

[8] S. B. Thrun. Efficient exploration in reinforcement learning. Technical report, Carnegie Mellon University, 1992.

[9] S. D. Whitehead. Complexity and cooperation in q-learning. *Proc. 8th International Conf. on Machine Learning*, pages 363–367, 1991.

8 Appendix: Proof of Theorem4.1

The intuition for the proof is that α determines the probability of choosing a 'deviant' action from π'. If the current state distribution is $d(s|\pi,\mu)$ when an action from π' is chosen, then the performance change from this deviant action will be proportional to the policy advantage. The first deviant action will be chosen when the state distribution is approximately $d(s|\pi,\mu)$, and in the worst case scenario, all following deviant actions could be chosen when the state distribution is arbitrarily different. The penalty for the approximation in the former and the worst-case situation in the latter will be approximately ε.

Proof. The expected advantage with respect to π_{new} is,

$$\begin{aligned} & E_{a\sim\pi_{\text{new}}(a|s)}\{A_\pi(s,a)\} \\ = & \sum_a((1-\alpha)\pi(a|s)+\alpha\pi'(a|s))A_\pi(s,a) \\ = & \alpha\sum\pi'(a|s)A_\pi(s,a) \\ = & \alpha E_{a\sim\pi'(a|s)}\{A_\pi(s,a)\}. \end{aligned}$$

For any timestep, the probability that we choose an action according to π' is α. Let c_t be the random variable indicating the number of actions chosen from π' before time t. In particular, while no action has been chosen from π' we have $c_t = 0$ and after one action has been chosen $c_t = 1$. Hence, $\Pr(c_t = 0) = (1-\alpha)^t$ and $\Pr(c_t \geq 1) = 1-(1-\alpha)^t$. For any π', let $d_t(s|\pi')$ be the probability that the state at time t is s while following π' under the start-state distribution μ. It follows that for all timesteps t:

$$\begin{aligned} & E_{a,s\sim\pi_{\text{new}}(a|s)d_t(s|\pi_{\text{new}})}A_\pi(s,a) \\ = & \alpha E_{a,s\sim\pi'(a|s)d_t(s|\pi_{\text{new}})}A_\pi(s,a) \\ = & \alpha\Pr(c_t=0)E_{a,s\sim\pi'(a|s)d_t(s|\pi_{\text{new}},c_t=0)}A_\pi(s,a) \\ & +\alpha\Pr(c_t\geq 1)E_{a,s\sim\pi'(a|s)d_t(s|\pi_{\text{new}},c_t\geq 1)}A_\pi(s,a) \\ \geq & \alpha(1-\Pr(c_t\geq 1))E_{a,s\sim\pi'(a|s)d_t(s|\pi_{\text{new}},c_t=0)}A_\pi(s,a) \\ & -\alpha\Pr(c_t\geq 1)\varepsilon \\ \geq & \alpha E_{a,s\sim\pi'(a|s)d_t(s|\pi_{\text{new}},c_t=0)}A_\pi(s,a)-2\alpha\Pr(c_t\geq 1)\varepsilon \\ = & \alpha E_{a,s\sim\pi'(a|s)d_t(s|\pi)}A_\pi(s,a)-2\alpha(1-(1-\alpha)^t)\varepsilon \end{aligned}$$

where we have used the definition of ε and $d_t(s|\pi_{\text{new}}, c_t = 0) = d_t(s|\pi)$.

By substitution and lemma 6.1, we have:

$$\begin{aligned} & \omega(\pi_{\text{new}})-\omega(\pi) \\ = & \frac{1}{1-\gamma}E_{a,s\sim\pi_{\text{new}}(a|s)d(s|\pi_{\text{new}})}A_\pi(s,a) \\ = & \sum_{t=0}^{\infty}\gamma^t E_{a,s\sim\pi_{\text{new}}(a|s)d_t(s|\pi_{\text{new}})}\{A_\pi(s,a)\} \\ \geq & \sum_{t=0}^{\infty}\gamma^t(\alpha\sum_{s,a}d_t(s|\pi)\pi'(a|s)A_\pi(s,a)-2\alpha(1-(1-\alpha)^t)\varepsilon) \\ = & \frac{\alpha}{(1-\gamma)}E_{a,s\sim\pi'(a|s)d(s|\pi)}A_\pi(s,a)-2\alpha\varepsilon\sum_{t=0}^{\infty}\gamma^t(1-(1-\alpha)^t) \\ = & \alpha\frac{\mathbb{A}_\pi(\pi')}{(1-\gamma)}-2\alpha\varepsilon(\frac{1}{1-\gamma}-\frac{1}{1-\gamma(1-\alpha)}). \end{aligned}$$

The result follows from simple algebra. □