Reinforcement Learning in POMDP's via Direct Gradient Ascent

Jonathan Baxter JONATHAN.BAXTER@ANU.EDU.AU
Peter L. Bartlett PETER.BARTLETT@ANU.EDU.AU
Research School of Information Sciences and Engineering, Australian National University.

Abstract

This paper discusses theoretical and experimental aspects of gradient-based approaches to the direct optimization of policy performance in controlled POMDPs. We introduce GPOMDP, a REINFORCE-like algorithm for estimating an approximation to the gradient of the average reward as a function of the parameters of a stochastic policy. The algorithm's chief advantages are that it requires only a single sample path of the underlying Markov chain, it uses only one free parameter $\beta \in [0,1)$, which has a natural interpretation in terms of bias-variance trade-off, and it requires no knowledge of the underlying state. We prove convergence of GPOMDP and show how the gradient estimates produced by GPOMDP can be used in a conjugate-gradient procedure to find local optima of the average reward.

1. Introduction

"Reinforcement learning" is used to describe the general problem of training an agent to choose its actions so as to increase its long-term average reward. The structure of the environment is typically not known explicitly, so the agent is forced to learn by interaction with the environment.

In value-function based approaches to reinforcement learning the agent tries to learn the value of each state, or possibly each state-action pair. It then chooses the action with the highest value according to its value function. If the value function is exact then this approach is known to lead to the optimal policy under quite general conditions (Sutton & Barto, 1998; Bertsekas & Tsitsiklis, 1996). However, for many real-world problems it is intractable to represent the value function exactly and the agent instead tries to select a good approximation to the value function from a restricted class (for example, a neural-network or radial-basis-function class). This approach has yielded some remarkable empirical successes in learning to play games, including checkers (Samuel, 1959), backgammon (Tesauro, 1992; Tesauro, 1994), and chess (Baxter et al., 1999a). Successes outside of the games domain include job-shop scheduling (Zhang & Dietterich, 1995), and dynamic channel allocation (Singh & Bertsekas, 1997).

While there are many algorithms for training approximate value functions (see (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998) for comprehensive treatments), with varying degrees of convergence guarantees, all these algorithms—and indeed the approximate value function approach itself—suffer from a fundamental limitation: the error they seek to minimize does not guarantee good performance from the resulting policy. More precisely, there exist infinite horizon Markov Decision Processes (MDPs) with the following properties. For all $\epsilon > 0$ there is an approximate value function V with

$$\max_i |V(i) - V^*(i)| = \epsilon, \tag{1}$$

where the max is over all states i and $V^*(i)$ is the true value of state i under the optimal policy. However, the greedy policy based on this approximate value function has expected discounted reward

$$\eta = \eta^* - \frac{2\alpha\epsilon}{1-\alpha}, \tag{2}$$

where η^* is the expected discounted reward of the optimal policy and $\alpha \in [0,1)$ is the discount factor (Bertsekas & Tsitsiklis, 1996; Singh, 1994). Thus, even accurate approximations to the optimal value function can generate bad greedy policies if α is close to 1.

Because Equation (2) also defines the *worst* expected discounted reward of any greedy policy derived from an approximate value function satisfying (1), it has sometimes been used as a *motivation* for using approximate value function techniques. However, there are two objections to this. The first is that most existing algorithms for training approximate value functions do not minimize the maximum norm between V and V^*, but typically some ℓ_2 norm. Secondly, even if these algorithms did minimize the maximum norm directly, the smallest achievable error ϵ will be so large in many problems of practical interest that the bound (2) will be useless. Put another way, if we can choose a V to make ϵ arbitrarily small in (1), then we are not really in an approximate value function setting in the first place.

The fact that the class of approximate value functions does not contain a value function with small approximation error does not preclude it from containing a value function whose greedy policy approaches or even equals the performance of the optimal policy. All that matters for the performance of the greedy policy is the relative ordering the approximate value function assigns to the successor states in each state. This motivates an alternative approach: instead of seeking to minimize (1) or an ℓ_2 variant, one should minimize some form of relative error between state values (Baird, 1993; Bertsekas, 1997; Weaver & Baxter, 1999). While this idea is promising, the approach we take in this paper is even more direct: search for a policy maximizing the expected discounted reward directly.

We can view the average reward (2) as a function $\eta(\theta)$ of $\theta \in \mathbb{R}^K$, where θ are the parameters of V. Provided the dependence of η on θ is differentiable, we can compute $\nabla\eta(\theta)$ and then take a small step in the gradient direction in order to increase the average reward. Under general assumptions, such an approach will converge to a local maximum of the average reward η. In general, a *greedy* policy based on $V(\theta)$ will give a non-differentiable $\eta(\theta)$. Thus, in this paper we consider stochastic policies that generate distributions over actions rather than a deterministic action.

We cast our results and algorithms in the formal framework of Partially Observable Markov Decision Processes (POMDPs). The advantage of this framework is that it models uncertainty both in the state-transitions of an agent and in the observations the agent receives. The first contribution of this paper is GPOMDP, an algorithm for computing an approximation, $\nabla_\beta\eta(\theta)$, to the true gradient $\nabla\eta(\theta)$, from a single sample path of a POMDP. The algorithm requires storage of only $2K$ real numbers—where K is the number of parameters in the policy—and needs no knowledge of the underlying state. The accuracy of the approximation $\nabla_\beta\eta(\theta)$ is controlled by a parameter $\beta \in [0,1)$ (a discount factor) and, in particular, the accuracy is controlled by the relationship between β and the mixing time of the Markov chain underlying the POMDP (loosely speaking the mixing time is the time needed to approach stationarity from an arbitrary starting state). $\nabla_\beta\eta(\theta)$ has the property that $\lim_{\beta\to 1}\nabla_\beta\eta(\theta) = \nabla\eta(\theta)$. However, the trade-off preventing the setting of β arbitrarily close to 1 is that the variance of the algorithm's estimates increase as β approaches 1. We prove convergence with probability 1 of GPOMDP.

The second contribution of this paper is CONJPOMDP, a conjugate-gradient based optimization procedure that utilizes the estimates generated by GPOMDP. The key difficulty in performing greedy stochastic optimization is knowing when to terminate a line search, since noisy estimates make it very difficult to locate a maximum. We solve this problem in CONJPOMDP by using gradient estimates to bracket the maximum, rather than value estimates.

Finally, we present the results of an experiment which illustrates the key ideas of the paper. All proofs have been omitted due to space constraints.

1.1 Related Work

The gradient approach to reinforcement learning was pioneered by Williams (Williams, 1992), who introduced the REINFORCE algorithm for estimating the gradient in *episodic* tasks, for which there is an identified recurrent state i^*, and the algorithm returns a gradient estimate each time i^* is entered. Williams showed that the expected value of this estimate is the gradient direction, in the case that the number of steps between visits to i^* is a constant. Other formulae for the performance gradient of a Markov Decision Process that rely on the existence of a recurrent state have been given in (Glynn, 1986; Cao & Chen, 1997; Cao & Wan, 1998; Fu & Hu, 1994), and for POMDPs in (Jaakkola et al., 1995). Williams' algorithm was generalized to the infinite-horizon setting in (Kimura et al., 1997) and to more general reward structures in (Marbach & Tsitsiklis, 1998). The "VAPS" algorithm described in (Baird & Moore, 1999) showed how the algorithms in (Marbach & Tsitsiklis, 1998) could be interpreted as combining both value-function and policy-gradient approaches. VAPS also relies on the existence of recurrent states to guarantee convergence. VAPS can be viewed broadly under the banner of "Actor-Critic" algorithms (Barto et al., 1983), which have more recently been investigated in (Singh et al., 1995; Sutton et al., 2000; Konda & Tsitsiklis, 2000).

Policy-gradient algorithms for which convergence results have been proved all rely on the existence of an identifiable recurrent state, and the variance of these algorithms is related to the time between visits to the recurrent state. Although the assumptions we make in this paper about the POMDP ensure that every state is recurrent, we would expect that as the size of the state space increases, there will be a corresponding increase in the expected time between visits to the identified recurrent state. Furthermore, the time between visits depends on the parameters of the policy, and states that are frequently visited for the initial value of the parameters may become very rare as performance improves. In addition, in an arbitrary POMDP it may be difficult to estimate the underlying states, and therefore to determine when the gradient estimate should be updated. Thus, a key advantage of GPOMDP is that its running time is not bounded by the requirement to visit recurrent states. Instead, it is bounded by the "mixing" time of the POMDP (loosely, the time needed to approach stationarity), which is always shorter than recurrence time and often substantially so.

Approximate algorithms for computing the gradient were also given in (Marbach & Tsitsiklis, 1998; Marbach, 1998), one that sought to solve the aforementioned recurrence problem by demanding only recurrence to one of a set of recurrent states, and another that abandoned recurrence and used discounting, which is closer in spirit to our algorithm.

2. The Mathematical Framework

Our setting is that of an agent taking actions in an environment according to a parameterized policy. The agent seeks to adjust its parameters in order to maximize the long-term average reward. We pursue a local approach: get the agent to compute the gradient of the average reward with respect to its parameters, and then adjust the parameters in the gradient direction. Formally, the most natural setting for this problem is that of Partially Observable Markov Decision Processes or POMDPs. For ease of exposition we consider finite POMDPs. General results in the continuous case can be found in (Baxter & Bartlett, 1999).

Specifically, assume that there are n states $\mathcal{S} = \{1, \ldots, n\}$ of the world (including the agent's states), N controls $\mathcal{U} = \{1, \ldots, N\}$ and M observations $\mathcal{Y} = \{1, \ldots, M\}$. For each state $i \in \mathcal{S}$ there is a corresponding reward $r(i)$. Each $u \in \mathcal{U}$ determines a stochastic matrix $P(u) = [p_{ij}(u)]$ where $p_{ij}(u)$ is the probability of making a transition from state i to state j give control u. For each state $i \in \mathcal{S}$, an observation $y \in \mathcal{Y}$ is generated independently according to a probability distribution $\nu(i)$ over observations in $\mathcal{Y}$. We denote the probability of observation y by $\nu_y(i)$. A *randomized policy* is simply a function μ mapping observations $y \in \mathcal{Y}$ into probability distributions over the controls $\mathcal{U}$. That is, for each observation y, $\mu(y)$ is a distribution over the controls in $\mathcal{U}$. Denote the probability under μ of control u given observation y by $\mu_u(y)$. In general, to perform optimally, the policy has to be a function of the entire history of observations, but this can be achieved by concatenating observations and treating the vector of observations as input to the policy. One could also consider policies that have memory, such as parameterized finite automata, but that is the subject of some of our ongoing research and is beyond the scope of the present paper.

To each randomized policy $\mu(\cdot)$ and observation distribution $\nu(\cdot)$, there corresponds a Markov chain in which state transitions are generated by first selecting an observation y in state i according to the distribution $\nu(i)$, then selecting a control u according to the distribution $\mu(y)$, and then generating a transition to state j according to the probability $p_{ij}(u)$. To parameterize these chains we parameterize the policies, so that μ now becomes a function $\mu(\theta, y)$ of a set of parameters $\theta \in \mathbb{R}^K$ as well as the observation y. The Markov chain corresponding to θ has state transition matrix $P(\theta) = [p_{ij}(\theta)]$ given by $p_{ij}(\theta) = \mathbf{E}_{y\sim\nu(i)}\mathbf{E}_{u\sim\mu(\theta,y)}p_{ij}(u)$. Throughout, we assume that these Markov chains satisfy the following assumption:

Assumption 1. *Each $P(\theta)$ has a unique stationary distribution $\pi(\theta) := [\pi(\theta, 1), \ldots, \pi(\theta, n)]'$ satisfying the* balance equations *$\pi'(\theta)P(\theta) = \pi'(\theta)$ (throughout π' denotes the transpose of π). The magnitudes of the rewards, $|r(i)|$, are uniformly bounded by $R < \infty$ for all states i.*

Our goal is to find a $\theta \in \mathbb{R}^K$ maximizing the *long-term average reward*:

$$\eta(\theta) := \lim_{T\to\infty} \frac{1}{T}\mathbf{E}_\theta \left[\sum_{t=1}^{T} r(i_t)\right].$$

where $\mathbf{E}_\theta$ denotes the expectation over all sequences $i_0, i_1, \ldots$, with transitions generated according to $P(\theta)$. Under our assumptions, $\eta(\theta)$ is independent of the starting state i_0 and is equal to:

$$\eta(\theta) = \sum_{i=1}^{n} \pi(\theta, i) r(i) = \pi'(\theta) r, \tag{3}$$

where $r = [r(1), \ldots, r(n)]'$ (Bertsekas, 1995).

In contrast to the average reward, many approximate value-function based algorithms such as TD(λ) seek to optimize with respect to the expected *discounted* reward, where the latter is defined by $\eta_\alpha(\theta) := \sum_i \pi_i J_\alpha(\theta, i) = \sum_i \pi_i \mathbf{E}_\theta \left[\sum_{t=0}^{\infty} \alpha^t r(i_t) | i_0 = i\right]$ (here $\alpha \in [0, 1)$ is the discount factor). In this case the discounted value of state i, $J_\alpha(\theta, i)$, does depend on the starting state i. A curious fact about the present setting is that optimizing the long-term average reward is the same as optimizing expected discounted reward, since $\eta_\alpha(\theta) = \eta(\theta)/(1 - \alpha)$ (Singh et al., 1994, Fact 7). So without loss of generality we can consider just average reward.

3. Gradient Ascent on $\eta(\theta)$

The approach taken to optimization of $\eta(\theta)$ in this paper is *gradient ascent*. That is, repeatedly compute $\nabla\eta(\theta)$ with respect to the parameters θ, and then take a step in the uphill direction: $\theta \leftarrow \theta + \gamma\nabla\eta(\theta)$, for some suitable step-size γ.

A straightforward calculation shows that

$$\nabla\eta = \nabla\pi' r = \pi'\nabla P \left[I - P + e\pi'\right]^{-1} r, \tag{4}$$

where $e\pi'$ is the square matrix with each row equal to the stationary distribution π' (Baxter & Bartlett, 1999). Note that (4) should be read as K equations, one for each of the partial derivatives $\partial/\partial\theta_i$. For POMDPs with a sufficiently small number of states (and known transition probabilities and observation probabilities), (4) could be solved exactly

to yield the precise gradient direction. This may be an interesting avenue for further investigation since POMDPs are generally intractable even for small numbers of states (Papadimitriou & Tsitsiklis, 1987). However, in general the transition and observation probabilities will be unknown, and the state-space too large for the matrix inversion in the right-hand-side of (4) to be feasible. Thus, for many problems of practical interest, (4) will be intractable and we will need to find some other way of computing the gradient. One approximate technique for doing this is presented in the next section.

4. Approximating the Gradient $\nabla\eta(\theta)$

In this section, we show that the gradient can be split into two components, one of which becomes negligible as a discount factor β approaches 1.

Recall the definition of the discounted value of state i, $J_\beta(\theta, i)$. Here $\beta \in [0, 1)$ is the discount factor. Write $J_\beta(\theta) = [J_\beta(\theta, 1), \ldots, J_\beta(\theta, n)]'$ or simply $J_\beta = [J_\beta(1), \ldots, J_\beta(n)]'$ when the dependence on θ is obvious.

Theorem 2. *For all $\theta \in \mathbb{R}^K$ and $\beta \in [0, 1)$,*

$$\nabla\eta = (1-\beta)\nabla\pi' J_\beta + \beta\pi'\nabla P J_\beta. \tag{5}$$

We shall see in the next section that the second term in (5) can be estimated from a single sample path of the POMDP. The following theorem shows that the first term in (5) becomes negligible as β approaches 1. Notice that this is not immediate from Theorem 2, since J_β can become arbitrarily large in the limit $\beta \to 1$.

Theorem 3. *For all $\theta \in \mathbb{R}^K$,*

$$\nabla\eta = \lim_{\beta\to 1} \nabla_\beta\eta, \tag{6}$$

where

$$\nabla_\beta\eta := \pi'\nabla P J_\beta. \tag{7}$$

Theorem 3 shows that $\nabla_\beta\eta$ is a good approximation to the gradient as β approaches 1, but it turns out that values of β very close to 1 lead to large variance in the estimates of $\nabla_\beta\eta$ that we describe in the next section (that is, the estimates produced by GPOMDP). However, the following theorem shows that $1-\beta$ need not be too small, provided the Markov chain corresponding to $P(\theta)$ has a short *mixing time*. From any initial state, the distribution over states of a Markov chain converges to the stationary distribution, provided Assumption 1 about the existence and uniqueness of the stationary distribution is satisfied (Lancaster & Tismenetsky, 1985 Second Edition, Theorem 15.8.1, p. 552).

To precisely quantify mixing time, let $\|p - q\|_1$ denote the usual ℓ_1 distance on distributions $p = (p_1, \ldots, p_n), q = (q_1, \ldots, q_n)$: $\|q - p\|_1 = \sum_{i=1}^n |q_i - p_i|$. Let $p^t(i)$ denote the distribution over the states of the Markov chain at time t, starting from state i. Define $d(t)$ by $d(t) := \max_{i,j\in\mathcal{S}} \|p^t(i) - p^t(j)\|_1$. Note that $d(t)$ is a function of the parameters θ via the transition matrix $P(\theta)$, and since the state distribution converges to $\pi(\theta)$ for each θ, $d(t)$ must converge to zero. Finally, define the *mixing time* $\tau^*(\theta)$ of the Markov chain by:

$$\tau^*(\theta) := \min\left\{t\colon d(t) \le e^{-1}\right\}. \tag{8}$$

Theorem 4. *There exists a universal constant $C = C(B, R, n)$ such that for all $\beta \in [0, 1)$ and $\theta \in \mathbb{R}^k$,*

$$\|\nabla\eta(\theta) - \nabla_\beta\eta(\theta)\| \le C\tau^*(\theta)(1-\beta), \tag{9}$$

where B and R are the bounds on $|\nabla\mu/\mu|$ and the rewards respectively, n is the number of states in the Markov chain, and the norm $\|\cdot\|$ is the usual two-norm.

Theorem 4 shows that provided $1/(1-\beta)$ is large compared with the mixing time $\tau^*(\theta)$, $\nabla_\beta\eta(\theta)$ will be a good approximation to $\nabla\eta(\theta)$. Of course, in general the mixing time $\tau^*(\theta)$ will be unknown, but the purpose of Theorem 4 is not so much to provide a prescription for choosing β, but to enhance our understanding of the role β plays in the accuracy of the approximation $\nabla_\beta\eta(\theta)$.

5. Estimating $\nabla_\beta\eta(\theta)$

Having shown that $\nabla_\beta\eta(\theta)$ can be made a sufficiently accurate approximation to $\eta(\theta)$ by choosing the discount factor β judiciously in relation to the mixing time of the underlying Markov chain, we now describe GPOMDP, an algorithm for estimating $\nabla_\beta\eta(\theta)$ from a single sample path of the POMDP. To understand the algorithm, recall that the POMDP iterates as follows: at time step t the environment is in some state which we denote by i_t. An observation y_t is generated according to the distribution $\nu(i_t)$. The agent generates a control u_t according to the distribution given by its policy $\mu(\theta, y_t)$. Finally, the environment makes a transition to a new state i_{t+1} according to the probability $p_{i_t i_{t+1}}(u_t)$. GPOMDP is described in Algorithm 1. Note that the update for Δ_t is recursively computing the average of $r(i_0)z_0 + \cdots + r(i_t)z_t$. We now show that the estimate Δ_t produced by GPOMDP at time step t converges to $\nabla_\beta\eta(\theta)$ as the running time t approaches infinity. For this we need one more assumption:

Assumption 5. *The derivatives, $\frac{\partial\mu_u(\theta,y)}{\partial\theta_k}$ exist for all $u \in \mathcal{U}$, $y \in \mathcal{Y}$ and $\theta \in \mathbb{R}^K$. The ratios*

$$\left[\frac{\left|\frac{\partial\mu_u(\theta,y)}{\partial\theta_k}\right|}{\mu_u(\theta,y)}\right]_{y=1\ldots M;u=1\ldots N;k=1\ldots K}$$

are uniformly bounded by $B < \infty$ for all $\theta \in \mathbb{R}^K$.

Algorithm 1 The GPOMDP algorithm.

1: **Given:**

- Parameterized class of randomized policies $\{\mu(\theta,\cdot) : \theta \in \mathbb{R}^K\}$ satisfying Assumption 5.
- Partially observable Markov decision process which when controlled by the randomized policies $\mu(\theta,\cdot)$ corresponds to a parameterized class of Markov chains satisfying Assumption 1.
- $\beta \in [0,1)$.
- Arbitrary (unknown) starting state i_0.
- Observation sequence $y_0, y_1, \ldots$ generated by the POMDP with controls $u_0, u_1, \ldots$ generated randomly according to $\mu(\theta, y_t)$.
- Bounded reward sequence $r(i_0), r(i_1), \ldots$, where $i_0, i_1, \ldots$ is the (hidden) sequence of states of the Markov decision process.

2: Set $z_0 = 0$ and $\Delta_0 = 0$ ($z_0, \Delta_0 \in \mathbb{R}^K$).
3: **for** each observation y_t, control u_t, and subsequent reward $r(i_{t+1})$ **do**
4: $\quad z_{t+1} = \beta z_t + \frac{\nabla \mu_{u_t}(\theta, y_t)}{\mu_{u_t}(\theta, y_t)}$
5: $\quad \Delta_{t+1} = \Delta_t + \frac{1}{t+1}\left[r(i_{t+1}) z_{t+1} - \Delta_t\right]$
6: **end for**

This assumption should not be surprising since $\nabla\mu/\mu$ appears in the update of z_t in GPOMDP. Since μ appears in the denominator, we require that if the probability goes to zero for some θ, then so too must the gradient (and at at least the same rate).

Theorem 6. *Under Assumptions 1 and 5, Algorithm 1 starting from any initial state i_0 will generate a sequence $\Delta_0, \Delta_1, \ldots, \Delta_t, \ldots$ satisfying*

$$\lim_{t\to\infty} \Delta_t = \nabla_\beta \eta \quad \text{w.p.1.} \tag{10}$$

Theorem 6 provides a characterization of the limiting behavior of GPOMDP. In fact, we also have a result characterizing the finite time behavior of GPOMDP. Loosely speaking, provided

$$t > \Omega\left(\frac{\tau^*(\theta)}{\epsilon^2(1-\beta)^2}\right), \tag{11}$$

then $\|\Delta_t - \nabla_\beta \eta(\theta)\|_\infty < \epsilon$ with high probability (see (Bartlett & Baxter, 2000) for a more precise statement). Comparing (11) and (9), we can see the bias/variance tradeoff inherent in the choice of β: equation (9) tells us that to reduce the bias in the estimate $\nabla_\beta \eta(\theta)$ we must set β close to 1, while (11) indicates that to reduce the variance in the estimates of $\nabla_\beta \eta(\theta)$ produced by GPOMDP at time t, we should set β as close to 0 as possible.

6. Stochastic Gradient Ascent Algorithms

One technique for optimizing POMDPs using GPOMDP would be to repeatedly compute $\Delta_T(\theta)$ (the estimate produced by GPOMDP after T iterations with policy parameters θ), and then update the parameters by $\theta \leftarrow \theta + \gamma\Delta_T(\theta)$ for a suitable step-size γ. However, since the number of iterations T needed to ensure low variance in the estimates Δ_T can be quite large, we would like to make more efficient usage of the estimates by searching for a maximum in the direction Δ_T. CONJPOMDP, described in Algorithm 2, is a version of the Polak-Ribiere conjugate-gradient algorithm (Fine, 1999, S5.5.2, for example) that is designed to operate using only noisy (and possibly) biased estimates of the gradient of the objective function (for example, the estimates Δ_T provided by GPOMDP). The argument s_0 to CONJPOMDP provides an initial step-size for GSEARCH. When $\|\mathrm{GRAD}(\theta)\|^2$ falls below the argument ϵ, CONJPOMDP terminates.

Algorithm 2 $\mathrm{CONJPOMDP}(\mathrm{GRAD}, \theta, s_0, \epsilon) \to \mathbb{R}^K$

1: **Given:**

- $\mathrm{GRAD}: \mathbb{R}^K \to \mathbb{R}^K$: an estimate of the gradient of the objective function to be maximized.
- Starting parameters $\theta \in \mathbb{R}^K$
- Initial step size $s_0 > 0$.
- Gradient resolution ϵ.

2: $g = h = \mathrm{GRAD}(\theta)$
3: **while** $\|g\|^2 \geq \epsilon$ **do**
4: $\quad \mathrm{GSEARCH}(\mathrm{GRAD}, \theta, h, s_0, \epsilon)$
5: $\quad \Delta = \mathrm{GRAD}(\theta)$
6: $\quad \gamma = (\Delta - g) \cdot \Delta / \|g\|^2$
7: $\quad h = \Delta + \gamma h$
8: $\quad$ **if** $h \cdot \Delta < 0$ **then**
9: $\quad\quad h = \Delta$
10: $\quad$ **end if**
11: $\quad g = \Delta$
12: **end while**
13: return θ

The linesearch algorithm GSEARCH (Algorithm 3) uses only gradient information to bracket the maximum, and then uses quadratic interpolation to jump to the maximum. To bracket the maximum in the direction θ^* from θ, GSEARCH finds two points θ_1 and θ_2 in that direction such that $\mathrm{GRAD}(\theta_1) \cdot \theta^* > 0$ and $\mathrm{GRAD}(\theta_2) \cdot \theta^* < 0$. This approach is far more robust than the use of function values. Even if the estimates $\mathrm{GRAD}(\theta)$ are noisy, the variance of $\mathrm{sign}\left[\mathrm{GRAD}(\theta_1) \cdot \theta^*\right]$ is independent of the distance between θ_1 and θ_2. (In contrast, the variance of a comparison of function values at two points increases as the points get

Algorithm 3 $\mathrm{GSEARCH}(\mathrm{GRAD}, \theta_0, \theta^*, s, \epsilon) \to \mathbb{R}^K$

Given:

- $\mathrm{GRAD}\colon \mathbb{R}^K \to \mathbb{R}^K$: gradient estimate.
- Starting parameters $\theta_0 \in \mathbb{R}^K$.
- Search direction $\theta^* \in \mathbb{R}^K$ with $\mathrm{GRAD}(\theta_0) \cdot \theta^* > 0$.
- Initial step size $s > 0$.
- Inner product resolution $\epsilon >= 0$.

$\theta = \theta_0 + s\theta^*$
$\Delta = \mathrm{GRAD}(\theta)$
if $\Delta \cdot \theta^* < 0$ **then**
  Step back to bracket the maximum:
  repeat
    $s_+ = s, p_+ = \Delta \cdot \theta^*, s = s/2$
    $\theta = \theta_0 + s\theta^*$
    $\Delta = \mathrm{GRAD}(\theta)$
  until $\Delta \cdot \theta^* > -\epsilon$
  $s_- = s$
  $p_- = \Delta \cdot \theta^*$
else
  Step forward to bracket the maximum:
  repeat
    $s_- = s, p_- = \Delta \cdot \theta^*, s = 2s$
    $\theta = \theta_0 + s\theta^*$
    $\Delta = \mathrm{GRAD}(\theta)$
  until $\Delta \cdot \theta^* < \epsilon$
  $s_+ = s$
  $p_+ = \Delta \cdot \theta^*$
end if
if $p_- > 0$ and $p_+ < 0$ **then**
  $s = \frac{s_- p_+ - s_+ p_-}{p_+ - p_-}$
else
  $s = \frac{s_- + s_+}{2}$
end if
return $\theta_0 + s\theta^*$

closer together.) The disadvantage is that it is not possible to detect extreme overshooting of the maximum using only gradient estimates. However, with careful control of the line search we did not find this to be a problem.

CONJPOMDP operates by iteratively choosing "uphill" directions and then searching for a local maximum in the chosen direction. In the rest of the paper, we assume that the GRAD argument to CONJPOMDP is GPOMDP.

7. Experiments

Due to space constraints we only have room to consider one experiment, and we have chosen a "toy" problem so that we can illustrate all the key ideas from the rest of the paper. Experiments closer to "reality" are discussed in (Baxter et al., 1999b).

Consider a three-state MDP, in each state of which there is a choice of two actions a_1 and a_2. Table 1 shows the transition probabilities as a function of the states and actions. Each state x has an associated two-dimensional feature vector $\phi(x) = (\phi_1(x), \phi_2(x))$, with values of $(12/18, 6/18), (6/18, 12/18), (5/18, 5/18)$ for each of A, B and C respectively. The reward is 1 in state C and 0 for the other two states. Clearly, the optimal policy is to always select the action that leads to state C with the highest probability, which from Table 1 means always selecting action a_2. This rather odd choice of feature vectors for the states ensures that a value function linear in those features and trained using $\mathrm{TD}(1)$—while observing the optimal policy—will implement a suboptimal one-step greedy lookahead policy itself (Weaver & Baxter, 1999). Thus, in contrast to the gradient based approach, for this system, $\mathrm{TD}(1)$ training a linear value function is guaranteed to produce a worse policy if it starts out observing the optimal policy.

Origin State	Action	Destination State Probabilities A	B	C
A	$a1$	0.0	0.8	0.2
A	$a2$	0.0	0.2	0.8
B	$a1$	0.8	0.0	0.2
B	$a2$	0.2	0.0	0.8
C	$a1$	0.0	0.8	0.2
C	$a2$	0.0	0.2	0.8

Table 1. Transition probabilities of the three-state MDP

7.1 Training a Controller

Our goal is to learn a stochastic controller for this system that implements an optimal (or near-optimal) policy. Given a parameter vector $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$, we generate a policy as follows. For any state x, let $s_1(x) := \theta_1\phi_1(x) + \theta_2\phi_2(x)$, and $s_2(x) := \theta_3\phi_1(x) + \theta_4\phi_2(x)$. The probability of choosing action a_1 in state x is given by $\mu_{a_1}(x) = \frac{e^{s_1(x)}}{e^{s_1(x)} + e^{s_2(x)}}$, with the probability of choosing action a_2 given by $1 - \mu_{a_1}(x)$. The ratios $\frac{\nabla\mu_{a_i}(x)}{\mu_{a_i}(x)}$ needed by Algorithms GPOMDP are given by,

$$\frac{\nabla\mu_{a_1}(x)}{\mu_{a_1}(x)} = \frac{e^{s_2(x)}}{e^{s_1(x)} + e^{s_2(x)}}\left[\phi_1(x), \phi_2(x), -\phi_1(x), -\phi_2(x)\right]$$
$$\frac{\nabla\mu_{a_2}(x)}{\mu_{a_2}(x)} = \frac{e^{s_1(x)}}{e^{s_1(x)} + e^{s_2(x)}}\left[-\phi_1(x), -\phi_2(x), \phi_1(x), \phi_2(x)\right].$$

Note that these controllers satisfy assumption 5.

7.2 Gradient Estimates

With a parameter vector[1] of $\theta = [1, 1, -1, -1]$, estimates Δ_T of $\nabla_\beta\eta$ were generated using GPOMDP, for various values of T and $\beta \in [0, 1)$. To measure the progress of Δ_T

[1]Other initial values of the parameter vector were chosen with similar results. Note that $[1, 1, -1, -1]$ generates a suboptimal policy.

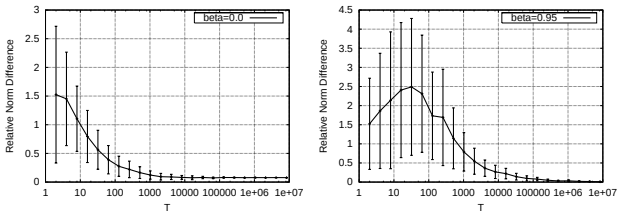

Figure 1. A plot of $\frac{\|\nabla\eta - \Delta_T\|}{\|\nabla\eta\|}$ for the three-state Markov chain, for two values of the discount parameter β.

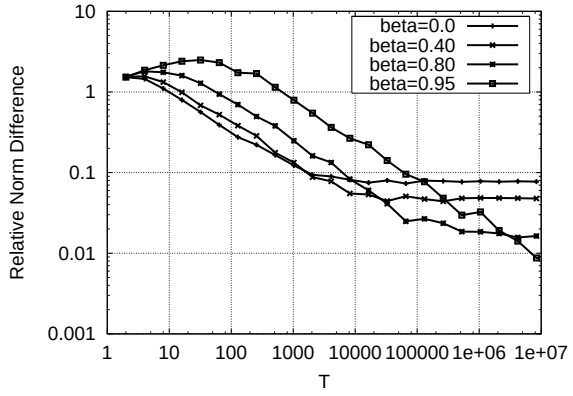

Figure 2. Graph showing the final bias in the estimate Δ_T (as measured by $\frac{\|\nabla\eta - \Delta_T\|}{\|\nabla\eta\|}$) as a function of β for the three-state Markov chain. Δ_T was generated by Algorithm 1. Note both axes are log scales.

towards the true gradient $\nabla\eta$, $\nabla\eta$ was calculated from (4) and then for each value of T the relative error $\frac{\|\Delta_T - \nabla\eta\|}{\|\nabla\eta\|}$ was recorded. The relative errors are plotted in 1 and 2. The first Figure shows how large β increases the variance of GPOMDP, while the second Figure shows a corresponding decrease in the final bias. Taken together they illustrate nicely the bias/variance trade-off in the choice of β.

7.3 Training via Conjugate-Gradient Ascent

CONJPOMDP with GPOMDP as the "GRAD" argument was used to train the parameters of the controller described in the previous section. Following the low bias observed in the experiments of the previous section, the argument β of GPOMDP was set to 0. After a small amount of experimentation, the arguments s_0 and ϵ of CONJPOMDP were set to 100 and 0.0001 respectively. None of these values were critical, although the extremely large initial step-size (s_0) did considerably reduce the time required for the controller to converge to near-optimality.

Figure 3 shows the average reward $\eta(\theta)$ of the final controller produced by CONJPOMDP, as a function of the total number of simulation steps of the underlying Markov chain. The plots represent an average over 500 independent runs of CONJPOMDP. Note that 0.8 is the average reward of the optimal policy. The parameters of the controller were (uniformly) randomly initialized in the range $[-0.1, 0.1]$ before each call to CONJPOMDP. After each call to CONJPOMDP, the average reward of the resulting controller was computed exactly by calculating the stationary distribution for the controller.

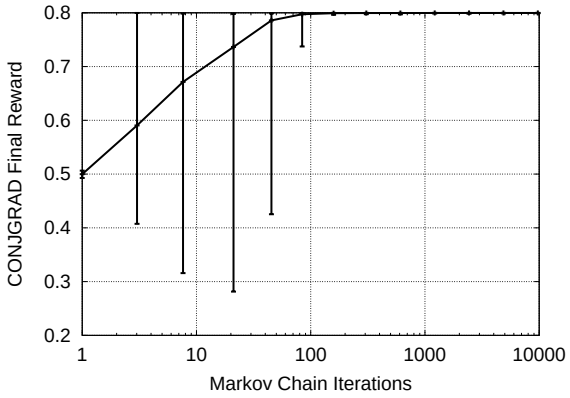

Figure 3. Performance of the 3-state Markov chain controller trained by CONJPOMDP as a function of the total number of iterations of the Markov chain.

8. Conclusion

Gradient-based approached to reinforcement learning hold much promise as a means to solve problems of partial observability and to avoid some of the pitfalls associated with non-convergence of value-function methods. A third reason for preferring direct policy approaches is that it is often far easier to construct a reasonable class of parameterized policies than it is to construct a class of value functions; we often know *how* to act without being able to compute the *value* of acting.

In this paper we have analyzed one algorithm for computing an approximation to the performance gradient. There should be many possible generalizations to other approximate algorithms. We also showed how the approximate gradients could be used robustly in greedy local search. One weakness of our algorithm is the need to specify running times and the discount factor β in advance. we are currently investigating automatic algorithms for finding these variables.

It is somewhat "folklore" in the Machine Learning community that gradient-based methods suffer from unacceptably large variance. The reasons for this conclusion are still not clear and warrant further investigation. There are also many avenues for further research. Particularly exciting is the generalization of GPOMDP to multi-agent settings, and implications for learning in biological neural networks (Bartlett & Baxter, 1999).

References

Baird, L., & Moore, A. (1999). Gradient Descent for General Reinforcement Learning. *Advances in Neural Information Processing Systems 11*. MIT Press.

Baird, L. C. (1993). *Advantage Updating* (Technical Report WL-TR-93-1146). Wright Patterson AFB OH.

Bartlett, P. L., & Baxter, J. (1999). *Hebbian Synaptic Modifications in Spiking Neurons that Learn* (Technical Report). Australian National University.

Bartlett, P. L., & Baxter, J. (2000). *Estimation and approximation bounds for gradient-based reinforcement learning* (Technical Report). Australian National University. http://csl.anu.edu.au/∼jon/papers/drl_colt00.ps.gz.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*, 834–846.

Baxter, J., & Bartlett, P. L. (1999). *Direct Gradient-Based Reinforcement Learning: I. Gradient Estimation Algorithms* (Technical Report). Research School of Information Sciences and Engineering, Australian National University.

Baxter, J., Tridgell, A., & Weaver, L. (1999a). Learning to Play Chess Using Temporal-Differences. *Machine Learning*, -. To appear.

Baxter, J., Weaver, L., & Bartlett, P. L. (1999b). *Direct Gradient-Based Reinforcement Learning: II. Gradient Descent Algorithms and Experiments* (Technical Report). Research School of Information Sciences and Engineering, Australian National University.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control, Vol II*. Athena Scientific.

Bertsekas, D. P. (1997). Differential Training of Rollout Policies. *Proceedings of the 35th Allerton Conference on Communication, Control, and Computing*. Allerton Park, Ill.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Cao, X.-R., & Chen, H.-F. (1997). Perturbation Realization, Potentials, and Sensitivity Analysis of Markov Processes. *IEEE Transactions on Automatic Control, 42*, 1382–1393.

Cao, X.-R., & Wan, Y.-W. (1998). Algorithms for Sensitivity Analysis of Markov Chains Through Potentials and Perturbation Realization. *IEEE Transactions on Control Systems Technology, 6*, 482–492.

Fine, T. L. (1999). *Feedforward Neural Network Methodology*. New York: Springer.

Fu, M. C., & Hu, J. (1994). Smooth Perturbation Derivative Estimation for Markov Chains. *Operations Research Letters, 15*, 241–251.

Glynn, P. W. (1986). Stochastic Approximation for Monte-Carlo Optimization. *Proceedings of the Winter Simulation Conference* (pp. 285–289).

Jaakkola, T., Singh, S. P., & Jordan, M. I. (1995). Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In G. Tesauro, D. Touretzky and T. Leen (Eds.), *Advances in neural information processing systems*, vol. 7. Cambridge, MA: MIT Press.

Kimura, H., Miyazaki, K., & Kobayashi, S. (1997). Reinforcement learning in POMDPs with function approximation. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)* (pp. 152–160).

Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-Critic Algorithms. *Neural Information Processing Systems 1999*. MIT Press. To Appear.

Lancaster, P., & Tismenetsky, M. (1985 (Second Edition)). *The Theory of Matrices*. San Diego, CA: Academic Press.

Marbach, P. (1998). *Simulation-Based Methods for Markov Decision Processes*. Doctoral dissertation, Labortory for Information and Decision Systems, MIT.

Marbach, P., & Tsitsiklis, J. N. (1998). *Simulation-Based Optimization of Markov Reward Processes* (Technical Report). MIT.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The Complexity of Markov Decision Processes. *Mathematics of Operations Research, 12*, 441–450.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development, 3*, 210–229.

Singh, S. (1994). An Upper Bound on the Loss from Approximate Optimal Value Functions. *Machine Learning, 16*, 227–233.

Singh, S., & Bertsekas, D. (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference* (pp. 974–980). MIT Press.

Singh, S., Jaakkola, T., & Jordan, M. (1995). Reinforcement learning with soft state aggregation. In G. Tesauro, D. Touretzky and T. Leen (Eds.), *Advances in neural information processing systems*, vol. 7. Cambridge, MA: MIT Press.

Singh, S. P., Jaakkola, T., & Jordan, M. I. (1994). Learning Without State-Estimation in Partially Observable Markovian Decision Processes. *Proceedings of the Eleventh International Conference on Machine Learning*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge MA: MIT Press. ISBN 0-262-19398-1.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Neural Information Processing Systems 1999*. MIT Press. To Appear.

Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. *Machine Learning, 8*, 257–278.

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation, 6*, 215–219.

Weaver, L., & Baxter, J. (1999). *Reinforcement Learning From State and Temporal Differences* (Technical Report). Department of Computer Science, Australian National University. http://csl.anu.edu.au/∼jon/papers/std_full.ps.gz.

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning, 8*, 229–256.

Zhang, W., & Dietterich, T. (1995). A reinforcement learning approach to job-shop scheduling. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1114–1120). Morgan Kaufmann.