

Localizing Policy Gradient Estimates to Action Transitions

Gregory Z. Grudic

GRUDIC@LINC.CIS.UPENN.EDU

Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA

Lyle H. Ungar

UNGAR@CIS.UPENN.EDU

Computer and Information Science, University of Pennsylvania, Philadelphia, PA

Abstract

Function Approximation (FA) representations of the state-action value function Q have been proposed in order to reduce variance in performance gradients estimates, and thereby improve performance of Policy Gradient (PG) reinforcement learning in large continuous domains (e.g., the *PIFA* algorithm of Sutton et al. (in press)). We show empirically that although PIFA converges significantly faster than traditional PG algorithms such as REINFORCE which directly sample Q (without using FA), FA representations of Q are *not* necessary to reduce variance in performance gradient estimates, and PG algorithms which use *selective* direct samples of Q can converge orders of magnitude faster than PIFA. We present a new PG algorithm, called Action Transition Policy Gradient (*ATPG*), which uses direct samples of Q and restricts estimates of the gradient to coincide with action transitions, thus obtaining *relative* value estimates of executing actions, without using FA representations of Q . We prove that ATPG gives an unbiased estimate of the performance gradient, and converges to an optimal policy under piece-wise continuity conditions on the policy and the state-action value function. Further, in an experimental comparison with PIFA and REINFORCE, ATPG always outperforms both algorithms, taking orders of magnitude fewer iterations to converge on all but very simple problems.

1. Introduction

A number of recent papers have re-addressed the Policy Gradient (*PG*) formulation of Reinforcement Learning (*RL*) (Baird & Moore, 1999; Baxter &

Bartlett, 1999; Sutton et al., in press; Konda & Tsitsiklis, in press). The reason for this renewed interest is threefold. First, the PG formulation uses a function approximation (*FA*) representation (e.g. Neural Networks, decision trees, etc.) of the agent's policy (i.e. mapping between state and action), which directly addresses the need for generalization in RL problems that have large continuous state spaces (Kaelbling et al., 1996). Second, PG algorithms learn by estimating the gradient of the agent's reward function with respect to the parameterization of the FA representation of the policy. The computational cost of this estimate is linear in the number of parameters describing the policy, which is in stark contrast to the exponential growth associated with traditional RL algorithms (Kaelbling et al., 1996), and makes the PG formulation very attractive for high dimensional problems (Grudic & Ungar, in press).

The third reason for renewed interest in PG algorithms is that they are provably convergent. Specifically, REINFORCE (Williams, 1992), which is one of the earliest examples of PG RL, as well as more recent examples (Sutton et al., in press; Konda & Tsitsiklis, in press), are all theoretically guaranteed of converging to locally optimal policies. Further, although PG algorithms such as VAPS (Baird & Moore, 1999) do not in general guarantee convergence to a local optimum policy, they nevertheless guarantee convergence of the algorithm to some (perhaps suboptimal) policy.

The PG formulation of RL represents the agent's policy as $\pi(s, a; \theta)$, which denotes the probability that the agent chooses action a in state s . The parameter vector θ represents all of the modifiable parameters of the chosen function approximation representation (e.g. the weights which are adjusted during learning in a Neural Network FA representation). A reward function $\rho(\theta)$ is then defined and learning is done by estimating the *performance gradient* $\partial\rho/\partial\theta$, which is used to modify the policy in a direction that increases

reward. Thus, the agent learns by first estimating this gradient $\widehat{\partial\rho/\partial\theta}$ for the current policy π , and then updating the policy parameters θ as follows:

$$\theta_{t+1} = \theta_t + \alpha \frac{\widehat{\partial\rho}}{\partial\theta} \quad (1)$$

where α is a small positive step size. In practice $\widehat{\partial\rho/\partial\theta}$ is calculated from estimates of the state action value function $Q^\pi(s, a)$, which is the value of executing action a in state s , under the current policy π .

Given equation (1), it is not difficult to observe that the efficiency of any specific PG algorithm is directly related to how effectively it can estimate the performance gradient $\partial\rho/\partial\theta$. In fact, although REINFORCE is known to give an unbiased estimate of $\partial\rho/\partial\theta$, the variance in this estimate is typically very large and as a result REINFORCE converges very slowly (Baird & Moore, 1999; Sutton et al., in press). This observation is further supported by the experimental results presented in this paper. The reason for this high variance is that $\widehat{\partial\rho/\partial\theta}$ is calculated in REINFORCE using *one* $Q^\pi(s, a)$ estimate per state visited and, as noted by Sutton et al. (in press), the key requirement of the PG formulation is that *relative* estimates of the value of executing two or more actions in each state are necessary for PG algorithms to converge. Therefore, an agent using REINFORCE must pass through the same set of states many times, and hope that its stochastic policy executes a sufficient number of different actions in each state to give a good estimate of the relative value of executing each action in each state, and thus obtain a good $\partial\rho/\partial\theta$ estimate.

Function approximation representations of $Q^\pi(s, a)$ have recently been proposed as one way of decreasing the variance in estimating $\partial\rho/\partial\theta$, and thus improving the rate of convergence of PG algorithms (Sutton et al., in press; Konda & Tsitsiklis, in press). The basic concept is to build function approximation representations $f(s, a) \approx Q^\pi(s, a)$, and then use these approximations instead of direct samples of $Q^\pi(s, a)$ to estimate $\widehat{\partial\rho/\partial\theta}$. In this way, the relative values of executing each action in each state visited by the agent can be used to update the gradient. In (Sutton et al., in press; Konda & Tsitsiklis, in press) it is shown that if $f(s, a)$ is restricted to be of a certain form defined by the chosen FA representation for $\pi(s, a; \theta)$, then the resulting PG algorithm is guaranteed to converge to a locally optimal policy.

If PG algorithms which use FA estimates of $Q^\pi(s, a)$ are to converge faster than those which do not, then the following condition must be satisfied: the number of samples of $\widehat{Q}^\pi(s, a)$ required to get sufficiently ac-

curate approximation $f(s, a)$ must be smaller than the number required to get a sufficiently accurate $\widehat{\partial\rho/\partial\theta}$ from direct sampling of $\widehat{Q}^\pi(s, a)$. Further, this condition must be satisfied more often than not. If, as the policy is changed via updates of its parameters θ (1), the actual function $Q^\pi(s, a)$ changes very little, then there is a good chance that the number of samples of $\widehat{Q}^\pi(s, a)$ required to update $f(s, a)$ will be small. However, this continuity condition of $Q^\pi(s, a)$ with respect to θ can not always be guaranteed. In practice small changes in policy (θ) can lead to large discontinuous changes in the value function. For example, consider the case where after updating its policy the agent collides with an obstacle for the first time, receiving a negative reward in an region of the state space where its current estimate $f(s, a)$ previously only assumed positive rewards (see Section 7.2 for a simple example of such an environment). In such cases, many samples $\widehat{Q}^\pi(s, a)$ may be required to update the approximation $f(s, a)$, and in fact the convergence of the algorithm may be much slower than the convergence of an algorithm which does not build a FA estimate of $Q^\pi(s, a)$.

This paper shows that there exist PG algorithms that construct estimates of $\widehat{\partial\rho/\partial\theta}$ directly from samples of $Q^\pi(s, a)$ which converge significantly faster than methods that first learn FA estimates of $Q^\pi(s, a)$ and then use these to estimate $\widehat{\partial\rho/\partial\theta}$.

2. The Proposed ATPG Algorithm

In this paper we present a new PG algorithm, the Action Transition Policy Gradient (ATPG), which has the unique property that it estimates the performance gradient using estimates of the *relative* value of each action with respect to the average value of all the actions. The ATPG performance gradient estimate is:

$$\frac{\widehat{\partial\rho}}{\partial\theta} = \sum_s \sum_a g_\pi(s, a) \left(\widehat{Q}^\pi(s, a) - \widehat{V}^\pi(s) \right) \quad (2)$$

where $g_\pi(s, a)$ is analytically derived from the policy, and $\widehat{V}^\pi(s)$ is an estimate of $\overline{V}^\pi(s)$ which is the average value of all actions; for a policy of M possible actions, the exact expression for $\overline{V}^\pi(s)$ is defined by:

$$\overline{V}^\pi(s) = \frac{1}{M} \sum_{j=1}^M Q^\pi(s, a_j) \quad (3)$$

The intuition behind the ATPG estimation is as follows. If the execution in state s of action a_j has better than average reward under π , then the policy gradient in (2) will update the parameters θ such that the probability of executing a_j in s is increased. Conversely, if

the execution in state s of action a_j has worse than average reward under π , then the policy gradient in (2) will update the parameters θ such that the probability of executing a_j in s is decreased.

The ATPG algorithm is further unique in that the term $Q^\pi(s, a) - \bar{V}^\pi(s)$ in (2) is estimated by directly sampling $\widehat{Q}^\pi(s, a)$ only when the agent changes actions. The ATPG gradient estimate is made by comparing the values of $\widehat{Q}^\pi(s_t, a_t)$ and $\widehat{Q}^\pi(s_{t+1}, a_{t+1})$ whenever $a_t \neq a_{t+1}$. This property makes ATPG significantly different from other PG algorithms such as VAPS (Baird & Moore, 1999) and REINFORCE (Williams, 1992), which also use direct samples of $\widehat{Q}^\pi(s, a)$, but which update the gradient estimate at every state the agent visits. The result of this indiscriminant sampling is a large variance in the gradient estimate. The selective sampling used by ATPG both reduces variance in the gradient estimate and saves computation because no change is made until $a_t \neq a_{t+1}$. The motivation behind this strategy is that when two different actions are executed in close proximity in state space (i.e. $s_{t+1} \approx s_t$), their relative values in this region can be directly estimated, and therefore the gradient estimate in this region will be in the direction of the more valuable of the two actions. Further, if all different possible action transitions occur near one another in state space, then the sum of the PG estimates made after each transition gives a gradient estimate in the direction of the most valuable action. Formal proof of this requires both $Q^\pi(s, a)$ and $\pi(s, a; \theta)$ to be locally continuous in s in all regions of $s \in S$ where action transitions occur. However, one key to the fast convergence property of the ATPG algorithm is that $Q^\pi(s, a)$ need not be continuous with respect to θ , making it insensitive to discontinuities in the value function as the policy changes.

In this paper, we prove that the ATPG algorithm converges to a locally optimal policy under appropriate piece-wise continuity conditions on the policy $\pi(s, a; \theta)$ and the state-action value function $Q^\pi(s, a)$. Further, we present theory showing that ATPG gives an unbiased estimate of the performance gradient $\partial\rho/\partial\theta$ based on the relative value of each action with respect to the average value of all actions. These theoretical results are supported by an experimental comparison of the ATPG algorithm with REINFORCE and with the Policy Iteration and Function Approximation *PIFA* algorithm proposed by Sutton et al. (in press). Our experiments indicate that the ATPG consistently outperforms PIFA, giving an order of magnitude faster convergence when the value function is highly nonlinear or discontinuous. Furthermore, PIFA consistently outperformed REINFORCE by at least an order of

magnitude in all experiments.

3. Policy Gradient RL Formulation

The Policy Gradient formulation used in this paper is based on the *Policy Gradient* Theorem of Sutton et al. (in press). We model the RL problem as a Markov Decision Process (MDP). Let the agent's state at time $t \in \{1, 2, \dots\}$ be given by $s_t \in S$, $S \subseteq \mathfrak{R}^N$. At each time step the agent chooses from one of $M > 1$ discrete actions $a_t \in A$ and receives a reward $r_t \in \mathfrak{R}$. The dynamics of the environment are characterized by transition probabilities $P_{ss'}^a = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$ and expected rewards $R_s^a = E\{r_{t+1} | s_t = s, a_t = a\}$, $\forall s, s' \in S, a \in A$. The policy followed by the agent is characterized by a parameter vector $\theta \in \mathfrak{R}^D$, and is defined by the probability distribution $\pi(s, a; \theta) = Pr\{a_t = a | s_t = s; \theta\}$, $\forall s \in S, a \in A$. We assume that $\pi(s, a; \theta)$ is differentiable with respect to θ .

The Policy Gradient Theorem is valid for both the average and discount reward formulations. Here we limit our discussion to the discount reward formulation, however, all the theory presented is valid for both. The discounted reward function starting from state s_0 is given by:

$$\rho(\pi) = E \left\{ \sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_0, \pi \right\} \quad (4)$$

Finally, the weighting of states visited is:

$$d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr \{s_t = s \mid s_0, \pi\} \quad (5)$$

Given these definitions, the exact expression of the policy gradient expression is given by:

$$\frac{\partial\rho}{\partial\theta} = \sum_s d^\pi(s) \sum_a \frac{\partial\pi(s, a; \theta)}{\partial\theta} Q^\pi(s, a) \quad (6)$$

As discussed in Sutton et al. (in press), we can equivalently write this as:

$$\frac{\partial\rho}{\partial\theta} = \sum_s d^\pi(s) \sum_a \frac{\partial\pi(s, a; \theta)}{\partial\theta} (Q^\pi(s, a) - \bar{V}^\pi(s)) \quad (7)$$

where, $\bar{V}^\pi(s)$ is the average value of a state under π as defined in (3). The reason that (7) is a valid estimate of the performance gradient is that the key to estimating it is in determining the *relative* value of actions in any given state, not the absolute. Therefore, as long as a constant value (which in the case of (7) is $\bar{V}^\pi(s)$) is subtracted from each $Q^\pi(s, a)$ at each state s , the

gradient is valid. In this paper, the performance gradient defined in (7) is used because it is easily calculated within the ATPG framework described next.

4. Action Transition Policy Gradient

4.1 ATPG Approximation

Whenever the agent changes the action being executed (i.e. $a_t \neq a_{t+1}$), we update the policy gradient using the following approximation:

$$P_t \equiv \left[\frac{\partial \pi(s_t, a_t; \theta)}{\partial \theta} q_t + \frac{\partial \pi(s_{t+1}, a_{t+1}; \theta)}{\partial \theta} q_{t+1} \right] \quad (8)$$

where, for M possible actions:

$$q_t = \left(\frac{1}{M} \right) \frac{\widehat{Q}^\pi(s_t, a_t) - \overline{Q}_t}{\pi(s_t, a_t; \theta) \pi(s_{t+1}, a_{t+1}; \theta)} \quad (9)$$

$$q_{t+1} = \left(\frac{1}{M} \right) \frac{\widehat{Q}^\pi(s_{t+1}, a_{t+1}) - \overline{Q}_t}{\pi(s_t, a_t; \theta) \pi(s_{t+1}, a_{t+1}; \theta)} \quad (10)$$

and $\widehat{Q}^\pi(s, a)$ is an unbiased estimate of $Q^\pi(s, a)$, and \overline{Q}_t is the average of the state-action value functions:

$$\overline{Q}_t = \frac{1}{2} \left(\widehat{Q}^\pi(s_t, a_t) + \widehat{Q}^\pi(s_{t+1}, a_{t+1}) \right) \quad (11)$$

The motivation behind the PG approximation defined by (8) through (11) is as follows. First, we are only interested in estimating the PG after an action transition (hence equation (8)). Second, we care only about the relative magnitude of $\widehat{Q}^\pi(s_t, a_t)$ and $\widehat{Q}^\pi(s_{t+1}, a_{t+1})$. Therefore we subtract the average value from each such that $q_t = -q_{t+1}$; the result of this shifting of the Q values is that the gradient estimate will move towards increasing the probability of executing the more valuable action. Third, normalizing by M and $\pi(s_t, a_t; \theta) \pi(s_{t+1}, a_{t+1}; \theta)$ in (9) and (10) accounts for the averaging over M possible actions and the probability of executing a_t and then a_{t+1} in a MDP.

We assume an unbiased estimate of the state action value function Q^π at the end of each episode is given by:

$$\widehat{Q}^\pi(s_t, a_t) = \sum_{k=1}^{H-t} \gamma^{k-1} r_{t+k} \quad (12)$$

where H is the number of time steps executed by the agent during the episode (i.e. $t = 1, \dots, H$), and r_t are the rewards received by the agent.

Given the PG approximation in (8), the estimate of the policy gradient after a single episode l is given by:

$$\left[\frac{\partial \rho}{\partial \theta} \right]_l = \sum_{t=1}^H \varphi_t \quad (13)$$

where

$$\varphi_t = \begin{cases} P_t & \text{if } a_t \neq a_{t+1} \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

If the estimate of the policy gradient is based on L episodes, then, the following ATPG Approximation is used:

$$\frac{\widehat{\partial \rho}}{\partial \theta} = \frac{1}{L} \sum_{l=1}^L \left[\frac{\widehat{\partial \rho}}{\partial \theta} \right]_l \quad (15)$$

To prove that the ATPG approximation gives an unbiased estimate of the state action value function, we make the following piece-wise Lipschitz smoothness assumption on Q^π :

$$\begin{aligned} & \forall s \in S, S \subseteq \mathbb{R}^N, a \in A \\ & \exists (k_1 > 0, k_1 \in \mathbb{R}), \exists (\delta \in \mathbb{R}^N), \exists (\epsilon \in \mathbb{R}, \epsilon > 0), \text{ s.t.} \\ & \quad \forall \|\delta\| \leq \epsilon \rightarrow s + \delta \in S \wedge \\ & \quad |Q^\pi(s, a) - Q^\pi(s + \delta, a)| \leq k_1 \|\delta\| \end{aligned} \quad (16)$$

In addition, we assume that $\pi(s, a; \theta)$ is piece-wise continuous with respect to s . Formally, this is defined as follows:

$$\begin{aligned} & \forall s \in S, S \subseteq \mathbb{R}^N, a \in A \\ & \exists (k_2 > 0, k_2 \in \mathbb{R}), \exists (\delta \in \mathbb{R}^N), \exists (\epsilon \in \mathbb{R}, \epsilon > 0), \text{ s.t.} \\ & \quad \forall \|\delta\| \leq \epsilon \rightarrow s + \delta \in S \wedge \\ & \quad |\pi(s, a; \theta) - \pi(s + \delta, a; \theta)| \leq k_2 \|\delta\| \end{aligned} \quad (17)$$

We now state the following theorem.

Theorem: ATPG Approximation *At each time t , let the step the agent takes be bounded by $(s_t - s_{t+1}) \leq \delta$ for $s, \delta \in \mathbb{R}^N$. Assume that Q^π satisfies the Lipschitz smoothness condition (16), that \widehat{Q}^π is an unbiased estimate of Q^π , and that π satisfies (17) and is continuous w.r.t. θ . Assume also that the frequency of states visited under π is governed by $d^\pi(s)$. Then, as L becomes large in (15), and as $\|\delta\| \rightarrow 0$:*

$$E \left[\frac{\widehat{\partial \rho}}{\partial \theta} \right] \rightarrow \frac{\partial \rho}{\partial \theta} \quad (18)$$

Proof: Consider any state $s \in S$ which the agent visits under the current policy π . In this state the agent takes an action a_i with probability $\pi(s, a_i; \theta)$. The agent then takes a step δ and executes action a_j with probability $\pi(s + \delta, a_j; \theta)$. Therefore, given that the process is Markov, the probability of executing action a_i and then a_j is given by $\pi(s, a_i; \theta) \pi(s + \delta, a_j; \theta)$. Let the contribution to the policy gradient approximation in (8) from executing the two actions a_i and then a_j (for $a_i \neq a_j$) be symbolized by $\widehat{pg}_{ij}(s)$. As L becomes large and the agent visits state s many times its expected value is:

$$E[\widehat{pg}_{ij}(s)] = {}^i P_j(s) \pi(s, a_i; \theta) \pi(s + \delta, a_j; \theta) \quad (19)$$

where ${}^i P_j(s)$ is derived from (8) and has the form:

$${}^i P_j(s) = \left[\frac{\partial \pi(s, a_i; \theta)}{\partial \theta} q_i + \frac{\partial \pi(s + \delta, a_j; \theta)}{\partial \theta} q_j \right] \quad (20)$$

and, because $E[\widehat{Q}^\pi(s, a)] = Q^\pi(s, a)$, $\forall s \in S, \forall a \in A$:

$$q_i = \left(\frac{1}{M} \right) \frac{Q^\pi(s, a_i) - \overline{Q_{ij}}}{\pi(s, a_i; \theta) \pi(s + \delta, a_j; \theta)} \quad (21)$$

$$q_j = \left(\frac{1}{M} \right) \frac{Q^\pi(s + \delta, a_j) - \overline{Q_{ij}}}{\pi(s, a_i; \theta) \pi(s + \delta, a_j; \theta)} \quad (22)$$

$$\overline{Q_{ij}} = \frac{1}{2} (Q^\pi(s, a_i) + Q^\pi(s + \delta, a_j)) \quad (23)$$

Let $\widehat{pg}(s)$ represent the policy gradient estimate at state s made using (15) as L becomes large, given that all possible combinations of a_i and a_j (i.e. $i, j \in \{1, \dots, M\}, i \neq j$) can occur at state s . As L becomes large the expected value of $\widehat{pg}(s)$ is therefore given by:

$$E[\widehat{pg}(s)] = \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M E[\widehat{pg}_{ij}(s)] \quad (24)$$

Substituting (19) and cancelling $\pi(s, a_i; \theta) \pi(s + \delta, a_j; \theta)$ gives:

$$E[\widehat{pg}(s)] = \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M {}^i \Omega_j(s) \quad (25)$$

where

$${}^i \Omega_j(s) = \frac{\partial \pi(s, a_i; \theta)}{\partial \theta} \left(\frac{Q^\pi(s, a_i) - \overline{Q_{ij}}}{M} \right) + \frac{\partial \pi(s + \delta, a_j; \theta)}{\partial \theta} \left(\frac{Q^\pi(s + \delta, a_j) - \overline{Q_{ij}}}{M} \right) \quad (26)$$

From assumptions (16) and (17), we know that as $\|\delta\| \rightarrow 0$, so do $Q(s + \delta, a_j) \rightarrow Q(s, a_j)$ and $\pi(s + \delta, a_j; \theta) \rightarrow \pi(s, a_j; \theta)$. Therefore,

$${}^i \Omega_j(s) \rightarrow \frac{\partial \pi(s, a_i; \theta)}{\partial \theta} \left(\frac{Q^\pi(s, a_i) - \overline{Q_{ij}}}{M} \right) + \frac{\partial \pi(s, a_j; \theta)}{\partial \theta} \left(\frac{Q^\pi(s, a_j) - \overline{Q_{ij}}}{M} \right) \quad (27)$$

and

$$\overline{Q_{ij}} \rightarrow \frac{1}{2} (Q^\pi(s, a_i) + Q^\pi(s, a_j)) \quad (28)$$

Summing (25) over $j = \{1, \dots, M\}, j \neq i$, collecting all terms $\frac{\partial \pi(s, a_i; \theta)}{\partial \theta}$, $i = \{1, \dots, M\}$, and using (27) and (28), we get:

$$E[\widehat{pg}(s)] \rightarrow \sum_{i=1}^M \frac{\partial \pi(s, a_i; \theta)}{\partial \theta} \Psi_i(s) \quad (29)$$

where setting by definition $Q_n \equiv Q^\pi(s, a_n), \forall n \in \{1, \dots, M\}$, and noting that i and j are symmetric in (25) (hence the 2 in front),

$$\begin{aligned} \Psi_i &= \frac{2}{M} \left[(M-1) Q_i - \frac{1}{2} \left((M-1) Q_i + \sum_{\substack{j=1 \\ j \neq i}}^M Q_j \right) \right] \\ &= \frac{2}{M} \left[\frac{M}{2} Q_i - \frac{1}{2} \sum_{j=1}^M Q_j \right] \\ &= Q_i - \frac{1}{M} \sum_{j=1}^M Q_j \\ &= Q_i - \overline{V^\pi}(s) \end{aligned} \quad (30)$$

where $\overline{V^\pi}(s)$ is define in (3).

The entire policy gradient is obtained by summing $\widehat{pg}(s)$ over all states s visited under π as scaled by $d^\pi(s)$ as follows (i.e. equation (15) given the frequency of states visited defined by $d^\pi(s)$):

$$E \left[\frac{\partial \widehat{p}}{\partial \theta} \right] \rightarrow \sum_s d^\pi(s) E[\widehat{pg}(s)] \quad (31)$$

Finally, substituting (29) we get:

$$E \left[\frac{\partial \widehat{p}}{\partial \theta} \right] \rightarrow \sum_s d^\pi(s) \sum_{i=1}^M \frac{\partial \pi(s, a_i; \theta)}{\partial \theta} (Q^\pi(s, a_i) - V^\pi(s))$$

□

4.2 Episodic ATPG Algorithm

The *ATPG Algorithm* is:

STEP 1: *Estimate Policy Gradient:* After L episodes, use (15) to estimate the Policy Gradient.

STEP 2: *Update Policy Parameters:* Update the policy parameters θ as follows:

$$\theta_{n+1} = \theta_n + \alpha \left(\frac{\widehat{\partial p}}{\partial \theta} \div \left\| \frac{\widehat{\partial p}}{\partial \theta} \right\| \right) \quad (32)$$

GO TO **STEP 1**.

This algorithm has three learning parameters: L which defines the number of episodes the agent uses to estimate the PG; α which is the gradient step size; and δ which is the step size taken by the agent. In theory, both α and δ must be small for the ATPG algorithm to converge to a local minimum. In the experimental presented in Section 7, we simply fix these at small positive values. However, the theory presented

in the previous section also states that L should be large. Therefore, one of the goals of this paper is to experimentally determine how large L needs to be in practice. We do this by observing the total number of episodes required for ATPG to converge as L is increased.

5. Episodic REINFORCE Algorithm

Our Episodic REINFORCE (Williams, 1992) implementation follows that of the ATPG implementation described above, with the exception that the estimate of the performance gradient in **STEP 1** is made using

$$\frac{\widehat{\partial \rho}}{\partial \theta} = \sum_{s \in S_L} \frac{\partial \pi(s, a_s; \theta)}{\partial \theta} \widehat{Q}^\pi(s, a_s) \frac{1}{\pi(s, a_s; \theta)}$$

where the sum is over all the states S_L visited by the agent during the L episodes, and a_s is the action executed in state s . This version of REINFORCE follows that suggested by Sutton et al. (in press).

6. Episodic PIFA Algorithm

Our implementation of the *Episodic PIFA algorithm* is defined as follows:

STEP 1: *Approximate the State-Action Value Function:* Sample the state-action value function for L episodes and use gradient descent as described in (Sutton et al., in press) to update the weights of $f_w(s, a)$.

STEP 2: *Estimate Policy Gradient:* Estimate the performance gradient using

$$\frac{\widehat{\partial \rho}}{\partial \theta} = \sum_{s \in S_L} \sum_{i=1}^M \frac{\partial \pi(s, a_i; \theta)}{\partial \theta} f_w(s, a_i)$$

where the outside sum is over all the states S_L visited by the agent during the L episodes.

STEP 3: *Update Policy Parameters:* Update the policy parameters θ using (32). GOTO **STEP 1**.

There are two important details to note about our implementation of PIFA. The results reported here do not use a separate function approximation representation for the value function which Sutton et al. (in press) suggest could improve convergence results. We were unable to find a single representation for the value function which consistently improved convergence on our examples. Second, the algorithm converged extremely slowly unless we initially set all the weights of $f^\pi(s, a)$ to zero.

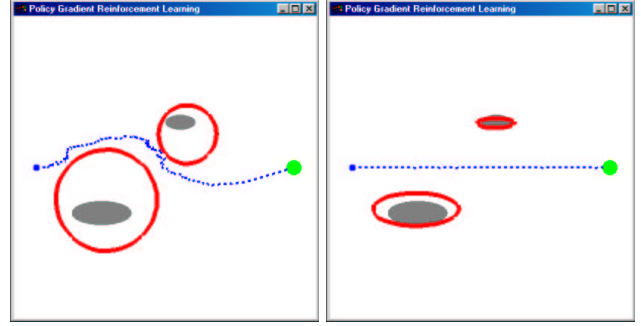


Figure 1. Initial and typical final policy after learning in the smooth value function simulation.

Our implementation of PIFA has four learning parameters. The first three it has in common with the ATPG algorithm: L , α , and δ . The additional learning parameter is the gradient parameter α_f which is used to update the weights of $f^\pi(s, a)$. As with the ATPG implementation, we simply fix α , δ and α_f to small positive values.

Finally, Sutton et al. (in press) give conditions under which $f^\pi(s, a)$ has converged given the current policy (see their equation (3)). Although this condition cannot be directly observed, theory implies that as L becomes large, the condition will be satisfied, much as the ATPG algorithm requires that L be large. Therefore, in experimentally comparing ATPG, PIFA and REINFORCE, we compare the number of episodes each takes to converge for different values of L .

7. Experimental Evaluation

We have simulated an agent executing a policy in a continuous two dimensional state space. The agent's policy is defined by Gaussians and the environment consists of obstacles, starting positions and a goal position. The agent receives a reward of +1 if the goal is reached, and a negative reinforcement of -1 each time it hits an obstacle. The agent's learning objective is thus to modify its policy such that it avoids hitting obstacles while minimizing the time it takes to reach the goal. The agent uses a discount reward formulation.

The agent can execute two types of actions: move away from the center of a Gaussian or move towards the center of a Gaussian. Thus the total number of actions is defined by the number of Gaussians in the agent's policy. There are four parameters per Gaussian: two defining its position (X, Y) in state space, and two defining its width (S_x, S_y) (these make up the θ parameters in the policy $\pi(s, a; \theta)$). At each location in state space, the agent chooses actions stochastically

based on the relative magnitude of each Gaussian.

Figure 1 depicts typical paths followed by the agent before and after learning as it moves along the dotted line from a starting point on the left hand side to the goal (small circle) on the right, while avoiding the obstacles (shaded ellipsoid region). The agent’s policy is defined by three Gaussians (i.e. 12 θ parameters define the policy $\pi(s, a; \theta)$). The location and width of two of the Gaussians is symbolized by regions enclosed by black elliptical curves, which represent areas of the state space where the action associated with the Gaussian has the greatest probability of being executed. The black ellipses symbolize the “move away from” the Gaussian center actions. The final Gaussian is centered at the goal position (the far right lightly shaded circle) and represents the “move towards” Gaussian center (i.e. goal position) action. The “move towards” goal action is most probably everywhere except within the black ellipse regions.

We compare the ATPG, PIFA and REINFORCE algorithms on two types of simulated environments: one depicted in Figure 1 which has a continuous value function with respect to changes in the policy (i.e. the function $Q^\pi(s, a)$ changes smoothly as the parameters θ of the policy change), and one depicted in Figure 2 which has a discontinuous value function with respect to changes in the policy.

7.1 Smooth Value Function Simulation

Figure 1 depicts the initial and learned policies of an agent in an environment where it never collides with an obstacle. Therefore as learning progresses and the policy $\pi(s, a; \theta)$ is changed, the value function $Q^\pi(s, a)$ changes continuously with respect to θ . The left graphic in Figure 1 shows a typical path given an initial policy specification, and the right graphic in Figure 1 shows a typical path taken under the learned policy after either the ATPG or PIFA algorithms have converged. Note that both algorithms converge to a policy which takes the agent from the initial position to the final position in the shortest number of steps.

The ATPG and PIFA algorithms were compared for $L = 1, 3, 5, 7,$ and 10 . The average number of episodes over ten runs for the ATPG algorithm to converge ranged from 120 (standard deviation 10) for $L = 1$, to 1300 (standard deviation 150) for $L = 10$. For the PIFA algorithm the number of episodes to converge ranged from 300 (standard deviation 20) for $L = 1$, to 5700 (standard deviation 980) for $L = 10$. Both algorithms obtained fastest convergence when the policy gradient was updated after each episode (i.e. $L = 1$). Therefore, for this smooth value function example, the

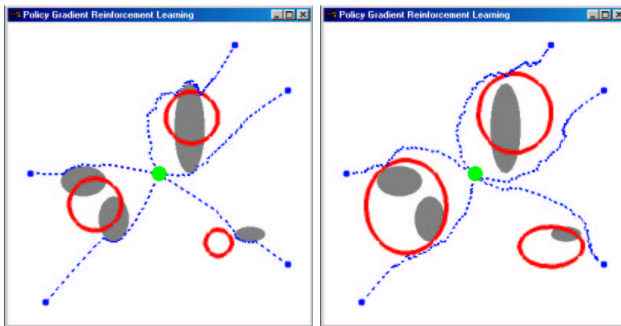


Figure 2. Initial and typical final policy after learning in the discontinuous value function simulation.

PIFA algorithm required on average about twice as many episodes to converge as the ATPG algorithm. Convergence of the REINFORCE algorithm over 10 consecutive trails was first observed for $L = 94$, on average taking 710,000 episodes (s. d. 60,000).

7.2 Discontinuous Value Function Simulation

The initial policy of the discontinuous value function (with respect to changes in the policy parameterization θ) simulation is shown in the left graphic of Figure 2. The policy is defined by four Gaussians (i.e. a total of 16 parameters) and the agent has five possible starting positions (shown as dots on the outside of the workspace) and one goal (shown as a shaded circle in the center). Note that each of the paths initially collide with obstacles, which implies that the value function will become discontinuous with respect to the parameterization of the policy when the agent learns to avoid the obstacles. A typical policy after either the ATPG or PIFA algorithms has converged is shown in right graphic of Figure 2. Note that this learned policy typically has no paths which collide with obstacles.

The ATPG and PIFA algorithms were compared for $L = 1, 3, 5, 7,$ and 10 . The average number of episodes over ten runs for the ATPG algorithm to converge ranged from 230 (standard deviation 30) for $L = 1$, to 2000 (standard deviation 290) for $L = 10$. For the PIFA algorithm the number of episodes to converge ranged from 2400 (standard deviation 340) for $L = 1$, to 13000 (standard deviation 3000) for $L = 10$. Both algorithms obtained fastest convergence when the policy gradient was updated after each episode (i.e. $L = 1$). Therefore, for this discontinuous value function example, the PIFA algorithm required on average about ten times as many episodes to converge as the ATPG algorithm. We did not observe convergence of the REINFORCE algorithm for this simulation. Our simulations were stopped at $L = 200$ and 1,000,000 episodes.

8. Discussion and Conclusion

We have presented a new policy gradient algorithm for reinforcement learning, Action Transition Policy Gradient (ATPG), which is unique in that it directly estimates the relative value of executing actions in a given state without using a function approximation representation of the value function. ATPG differs from other algorithms in that the gradient is updated only when an estimate of the *relative* value of two actions is observable, which happens whenever the agent changes actions. We have proven that ATPG is guaranteed to converge to a locally optimal policy under appropriate piece-wise continuity conditions on the policy and the state-action value function with respect to state.

An experimental comparison of the ATPG algorithm with the Policy Iteration for Function Approximation (PIFA) algorithm of Sutton et al. (in press) shows that on a simple learning scenario which has a value function that is smooth with respect to small policy changes, the PIFA algorithm took twice the number of episodes to converge as compared to the ATPG algorithm. In a slightly more complicated scenario which has a value function that is discontinuous with respect to small changes in the policy, the PIFA algorithm took ten times the number of episodes to converge as the ATPG algorithm. Further, in many more complicated scenarios (not reported here) where the ATPG algorithm typically converged in less than 300 episodes, the PIFA algorithm had still not converged after learning was stopped at 100,000 episodes. In fact, in all the simulated scenarios we have studied, ATPG converged in at most half the number of episodes for simple problems, and for more complicated problems typically converge in many orders of magnitude fewer episodes. It is also of note that when REINFORCE is applied to these same problems, it consistently requires at least 10 times more iterations than *either* ATPG or PIFA to converge.

We have applied the ATPG algorithm to many complicated reinforcement learning scenarios of the type presented above. In all cases we observed convergence in a reasonable number of episodes, even when the value function was highly discontinuous with respect to small changes in the policy. The fact that the rate of convergence of the ATPG algorithm is not dependent on the state-action value function being continuous with respect to the parameters of the policy is key to these fast convergence results. The scenarios we have studied are typical of robotics problems where small changes in policy can cause a robot to collide with an obstacle for the first time, thus resulting in a discontinuous change in the value function.

Our results have broader implications. In the policy gradient formulation of reinforcement learning, the knowledge the agent uses to make decisions is directly coded in the parameterized policy function. The key to getting effective learning within this framework lies in the agent's ability to obtain good estimates of the performance gradient with respect to the policy parameters. If there exists a direct sampling method for obtaining fast accurate estimates of this gradient without using function approximation (e.g., like the ATPG algorithm presented here), then using function approximation for the state-action value function is unnecessary, and in fact may be detrimental.

Acknowledgments

Thanks to Vijay Kumar and Jane Mulligan for valuable discussions. This work was funded by the IRCS at the University of Pennsylvania, and by the DARPA ITO MARS grant no. DABT63-99-1-0017.

References

- Baird, L., & Moore, A. W. (1999). Gradient descent for general reinforcement learning. *Advances in Neural Information Processing Systems 11*. Cambridge, MA: MIT Press.
- Baxter, J., & Bartlett, P. L. (1999). *Direct gradient-based reinforcement learning: I. gradient estimation algorithms* (Technical Report). Computer Sciences Laboratory, Australian National University.
- Grudic, G. Z., & Ungar, L. H. (in press). Localizing search in reinforcement learning. *Proceedings of the Seventeenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press / Cambridge, MA: MIT Press.
- Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237–285.
- Konda, V. R., & Tsitsiklis, J. N. (in press). Actor-critic algorithms. *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press.
- Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (in press). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 229–256.