Appears in Proc. of the 35th Allerton Conference on Communication, Control, and Computing, Allerton Park, Ill., October 1997

DIFFERENTIAL TRAINING OF[1] ROLLOUT POLICIES

by

Dimitri P. Bertsekas[2]

Abstract

We consider the approximate solution of stochastic optimal control problems using a neuro-dynamic programming/reinforcement learning methodology. We focus on the computation of a rollout policy, which is obtained by a single policy iteration starting from some known base policy and using some form of exact or approximate policy improvement. We indicate that, in a stochastic environment, the popular methods of computing rollout policies are particularly sensitive to simulation and approximation error, and we present more robust alternatives, which aim to estimate relative rather than absolute Q-factor and cost-to-go values. In particular, we propose a method, called *differential training*, that can be used to obtain an approximation to cost-to-go differences rather than cost-to-go values by using standard methods such as TD(λ) and λ-policy iteration. This method is suitable for recursively generating rollout policies in the context of simulation-based policy iteration methods.

1. INTRODUCTION

Consider the basic form of a dynamic programming problem, where we have the stationary discrete-time dynamic system

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, \tag{1}$$

where x_k is the state taking values in some set, u_k is the control to be selected from a finite set $U(x_k)$, and w_k is a random disturbance. We assume that the disturbances w_k, $k = 0, 1, \ldots$, are independent, and take values in a countable set. The one-stage cost function is denoted by $g(x, u, w)$. Also for notational simplicity, we suppress time indexes for f, g, and U, but our discussion fully applies to a time-varying context where f, g, and U depend on the time index k.

[1] Research supported by NSF under Grant 9625489-DMI. Thanks are due to John Tsitsiklis who suggested the analysis given at the end of Section 2.

[2] Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass., 02139.

We are given a policy $\pi = \{\mu_0, \mu_1, \ldots\}$, which at time k maps a state x to a control $\mu_k(x) \in U(x)$. We focus primarily on an N-stage horizon problem where k takes the values $0, 1, \ldots, N-1$, and there is a terminal cost $G(x_N)$ that depends on the terminal state x_N, but our discussion applies with minor modifications to various types of infinite horizon problems. The cost-to-go of π starting from an state x_k at time k will be denoted by

$$J_k(x_k) = E\left\{ G(x_N) + \sum_{i=k}^{N-1} g\big(x_i, \mu_i(x_i), w_i\big) \right\}. \tag{2}$$

The cost-to-go functions J_k satisfy the following recursion of dynamic programming (DP for short)

$$J_k(x) = E\big\{ g\big(x, \mu_k(x), w\big) + J_{k+1}\big(f\big(x, \mu_k(x), w\big)\big)\big\}, \qquad k = 0, 1, \ldots \tag{3}$$

with the initial condition

$$J_N(x) = G(x).$$

Our discussion applies with minor modifications to infinite horizon problems. For such problems, we need to assume that the given policy is stationary of the form $\pi = \{\mu, \mu, \ldots\}$, in which case the DP algorithm is replaced by its asymptotic form (Bellman's equation). For example, if the problem is discounted with discount factor $\alpha \in (0,1)$, the analog of Eq. (3) is

$$J(x) = E\big\{ g\big(x, \mu(x), w\big) + \alpha J\big(f\big(x, \mu(x), w\big)\big)\big\}, \qquad \forall\ x,$$

where $J(x)$ is the cost-to-go of π starting from x.

The *rollout policy based on* π is denoted by $\overline{\pi} = \{\overline{\mu}_0, \overline{\mu}_1, \ldots\}$, and is defined through the operation

$$\overline{\mu}_k(x) = \arg \min_{u \in U(x)} E\big\{ g(x, u, w) + J_{k+1}\big(f(x, u, w)\big)\big\}, \qquad \forall\ x,\ k = 0, 1, \ldots. \tag{4}$$

Thus the rollout policy is a one step-lookahead policy, with the optimal cost-to-go approximated by the cost-to-go of the base policy. The name "rollout policy" was introduced by Tesauro [TeG96] in connection with one of his simulation-based computer backgammon algorithms. The book by Bertsekas and Tsitsiklis [BeT96] discusses in some detail various aspects of rollout policies in a stochastic context, and also in a deterministic combinatorial optimization context, as a device for magnifying the effectiveness of heuristics (see also Bertsekas, Tsitsiklis, and Wu [BTW97]).

A standard policy iteration argument can be used to show that the rollout policy $\overline{\pi}$ is an improved policy over the base policy π. In particular, let $\overline{J}_k(x)$ be the cost-to-go of the rollout policy $\overline{\pi}$ starting from a state x_k at time k. It can be shown that $\overline{J}_k(x) \leq J_k(x)$ for all x and k, so that the rollout policy $\overline{\pi}$ is an improved policy over the base policy π. The proof uses backwards induction on k. In particular, we have $\overline{J}_N(x) = J_N(x) = G(x)$ for all x. Assuming that $\overline{J}_{k+1}(x) \leq J_{k+1}(x)$ for all x, we have for all x,

$$\begin{aligned}\overline{J}_k(x) &= E\big\{g\big(x,\overline{\mu}_k(x),w\big) + \overline{J}_{k+1}\big(f\big(x,\overline{\mu}_k(x),w\big)\big)\big\}\\ &\leq E\big\{g\big(x,\overline{\mu}_k(x),w\big) + J_{k+1}\big(f\big(x,\overline{\mu}_k(x),w\big)\big)\big\}\\ &\leq E\big\{g\big(x,\mu_k(x),w\big) + J_{k+1}\big(f\big(x,\mu_k(x),w\big)\big)\big\}\\ &= J_k(x),\end{aligned}$$

where the first inequality follows from the induction hypothesis, the second inequality follows from Eq. (4), and the final equality follows from the DP equation (3). This completes the induction proof that $\overline{\pi}$ is an improved policy over π.

Rollout policies are useful in two main contexts:

(a) *The one-time policy improvement context.* Here π is some "easily" implementable heuristic policy. By this we mean that the controls $\mu_k(x)$ can be easily calculated for every state x through some known algorithm. Frequently the rollout policy is a considerable improvement over the base policy, so if π is already a fairly good heuristic policy, the rollout policy can be pretty close to being optimal and can form the basis for effective suboptimal control.

(b) *The policy iteration context.* Here the policy improvement operation is used multiple times to generate a sequence of policies, and aims at obtaining an optimal policy (or a close approximation thereof).

The one-time policy improvement approach is particularly useful in an *on-line control context*, where the problem is not fully known in advance, but rather it is only known to belong to a fairly broad problem class. The problem becomes fully specified to the controller just before the first control u_0 is to be chosen. In this case, it is difficult to use a policy iteration approach, but a one-time improvement using a rollout policy may be possible, as long as there is a base/heuristic policy that works for the entire problem class.

In practice, one typically has a method or algorithm to compute the control $\mu_k(x)$ of the base policy, given the state x, but the corresponding cost-to-go functions J_k may not be known in close form. Then the exact or approximate computation of the rollout control $\overline{\mu}_k(x)$ using Eq. (4) becomes an important and nontrivial issue, since we need for all $u \in U(x)$ the value of

$$Q_k(x,u) = E\big\{g(x,u,w) + J_{k+1}\big(f(x,u,w)\big)\big\}, \tag{5}$$

known as the *Q-factor* at time k. Alternatively, for the computation of $\overline{\mu}_k(x)$ we need the value of the cost-to-go $J_{k+1}\big(f(x,u,w)\big)$ at all possible next states $f(x,u,w)$.

We are interested in methods for approximately computing $\overline{\pi}$ in highly complex large-scale problems. Our assumption is that we can simulate the system under the base policy π, and in particular, that we can generate sample system trajectories and corresponding costs consistently with the probabilistic data of the problem. The standard methods of computing $\overline{\pi}$ involve the use of a function $\tilde{J}_k(x, r)$ approximating $J_k(x)$, where r is a tunable parameter vector that is obtained by various training methods. This is the Neuro-Dynamic Programming/Reinforcement Learning (NDP/RL for short) approach (see the survey by Barto, Bradtke, and Singh [BBS95], and the research monograph by Bertsekas and Tsitsiklis [BeT96], which list many references). A number of training methods are described in these references, including the TD(λ) method of Sutton [Sut88], and the Q-learning method of Watkins [Wat89]. There are rigorous convergence results for these methods, due to Tsitsiklis [Tsi94], and Tsitsiklis and Van Roy [TsV96], and described in [BeT96].

We argue, however, in this paper that the standard NDP/RL training techniques lack robustness to approximation and simulation error, and that this is a major difficulty in practical computation. Similar concerns have been voiced by Werbos [Wer92a], [Wer92b], who has proposed several algorithimc ideas in a variety of contexts, including continuous-time deterministic optimal control. The proposal of advantage updating by Baird, Harmon, and Klopf [Bai93], [HBK94], is similarly motivated by the sensitivity to simulation and approximation error in the computation of the rollout policy. The main idea of advantage updating is to compute an approximation $\tilde{A}_k(x, u, r)$ to the *advantage function*

$$A_k(x,u) = Q_k(x,u) - \min_{u \in U(x)} Q_k(x,u')$$

corresponding to state control pairs (x, u), where r is a tunable parameter vector. The (approximate) rollout control at state x is then computed by minimizing $\tilde{A}_k\big(x, u, r\big) \geq 0$ over $u \in U(x)$. One of the difficulties with advantage updating is that effective training methods such as TD(λ) cannot be used in a straightforward fashion to train the advantage function approximation $\tilde{A}_k(x, u, r)$.

In this paper, we discuss two methods for reducing the effects of the simulation and approximation error. Both of these methods may be viewed as variance reduction techniques adapted to the NDP/RL setting. In the first method, we consider the evaluation of Q-factor differences by Monte-Carlo simulation. In the second method, which we call *differential training*, we construct a function approximation $\tilde{G}_k(x, x', r)$ to the cost-to-go difference $J_k(x) - J_k(x')$, where x and x' is any pair of states, and r is a tunable parameter vector. An interesting aspect of differential training is that, contrary to advantage updating, it can use any of the standard NDP/RL methods, including TD(λ). This is accomplished by training on a special *differential system*, whose state is a pair of states (x, x') of the original system.

2. EVALUATING Q-FACTORS BY SIMULATION

A conceptually straightforward approach to compute the rollout control at a given state x and time k is to use Monte-Carlo simulation (this was Tesauro's original proposal in the context of backgammon [TeG96]). To implement this, we consider all possible controls $u \in U(x)$ and we generate a "large" number of simulated trajectories of the system starting from x, using u as the first control, and using the policy π thereafter. Thus a simulated trajectory has the form

$$x_{i+1} = f\big(x_i, \mu_i(x_i), w_i\big), \qquad i = k+1, \ldots, N-1,$$

where the first generated state is

$$x_{k+1} = f(x, u, w_k),$$

and each of the disturbances $w_k, \ldots, w_{N-1}$ is an independent random sample from the given distribution. The costs corresponding to these trajectories are averaged to compute an approximation $\tilde{Q}_k(x,u)$ to the Q-factor

$$Q_k(x,u) = E\big\{g(x,u,w) + J_{k+1}\big(f(x,u,w)\big)\big\}. \tag{6}$$

Note here that $\tilde{Q}_k(x,u)$ is an approximation to $Q_k(x,u)$ because of the simulation error resulting from the use of a limited number of trajectories. The approximation becomes increasingly accurate as the number of simulated trajectories increases. Once the approximate Q-factor $\tilde{Q}_k(x,u)$ corresponding to each control $u \in U(x)$ is computed, we can obtain the (approximate) rollout control $\tilde{\mu}_k(x)$ by the minimization

$$\tilde{\mu}_k(x) = \arg \min_{u \in U(x)} \tilde{Q}_k(x,u). \tag{7}$$

There is a serious flaw with this approach, due to the simulation error involved in the calculation of the Q-factors. In particular, for the calculation of $\tilde{\mu}_k(x)$ to be accurate, the Q-factor differences

$$Q_k(x,u) - Q_k(x,\hat{u})$$

must be computed accurately for all pairs of controls u and $\hat{u}$, so that these controls can be accurately compared. On the other hand, the simulation/approximation errors in the computation of the individual Q-factors $Q_k(x,u)$ are magnified through the preceding differencing operation.

An alternative idea is to approximate by simulation the Q-factor difference $Q_k(x,u) - Q_k(x,\hat{u})$ by sampling the difference

$$C_k(x,u,\mathbf{w}_k) - C_k(x,\hat{u},\mathbf{w}_k),$$

where $\mathbf{w}_k = (w_k, w_{k+1}, \ldots, w_{N-1})$ and

$$C_k(x, u, \mathbf{w}_k) = G(x_N) + g(x, u, w_k) + \sum_{i=k+1}^{N-1} g\bigl(x_i, \mu_i(x_i), w_i\bigr). \tag{8}$$

This approximation may be far more accurate than the one obtained by differencing independent samples of $C_k(x, u, \mathbf{w}_k)$ and $C_k(x, \hat{u}, \mathbf{w}_k)$. Indeed, by introducing the zero mean sample errors

$$D_k(x, u, \mathbf{w}_k) = C_k(x, u, \mathbf{w}_k) - Q_k(x, u),$$

it can be seen that the variance of the error in estimating $Q_k(x, u) - Q_k(x, \hat{u})$ with the former method will be smaller than with the latter method if and only if

$$E_{\mathbf{w}_k,\, \hat{\mathbf{w}}_k} \left\{ \bigl| D_k(x, u, \mathbf{w}_k) - D_k(x, \hat{u}, \hat{\mathbf{w}}_k) \bigr|^2 \right\} > E_{\mathbf{w}} \left\{ \bigl| D_k(x, u, \mathbf{w}_k) - D_k(x, \hat{u}, \mathbf{w}_k) \bigr|^2 \right\},$$

or equivalently

$$E\bigl\{ D_k(x, u, \mathbf{w}_k) D_k(x, \hat{u}, \mathbf{w}_k) \bigr\} > 0; \tag{9}$$

that is, if and only if there is positive correlation between the errors $D_k(x, u, \mathbf{w}_k)$ and $D_k(x, \hat{u}, \mathbf{w}_k)$ (this idea was suggested by John Tsitsiklis). A little thought should convince the reader that this property is likely to hold in many types of problems. Roughly speaking, the relation (9) holds if changes in the value of u (at the first stage) have little effect on the value of the error $D_k(x, u, \mathbf{w}_k)$ relative to the effect induced by the randomness of $\mathbf{w}_k$. In particular, suppose that there exists a scalar $\gamma < 1$ such that, for all x, u, and $\hat{u}$, there holds

$$E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) - D_k(x, \hat{u}, \mathbf{w}_k) \bigr|^2 \right\} \le \gamma E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 \right\}. \tag{10}$$

Then we have

$$\begin{aligned} D_k(x, u, \mathbf{w}_k) D_k(x, \hat{u}, \mathbf{w}_k) &= \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 + D_k(x, u, \mathbf{w}_k)\bigl(D_k(x, \hat{u}, \mathbf{w}_k) - D_k(x, u, \mathbf{w}_k) \bigr) \\ &\ge \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 - \bigl| D_k(x, u, \mathbf{w}_k) \bigr| \bigl| D_k(x, \hat{u}, \mathbf{w}_k) - D_k(x, u, \mathbf{w}_k) \bigr|, \end{aligned}$$

from which we obtain

$$\begin{aligned} E\bigl\{ D_k(x, u, \mathbf{w}_k) D_k(x, \hat{u}, \mathbf{w}_k) \bigr\} &\ge E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 \right\} \\ &\quad - E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) \bigr| \bigl| D_k(x, \hat{u}, \mathbf{w}_k) - D_k(x, u, \mathbf{w}_k) \bigr| \right\} \\ &\ge E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 \right\} \\ &\quad - \frac{1}{2} E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 \right\} - \frac{1}{2} E \left\{ \bigl| D_k(x, \hat{u}, \mathbf{w}_k) - D_k(x, u, \mathbf{w}_k) \bigr|^2 \right\} \\ &\ge \frac{1 - \gamma}{2} E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 \right\}. \end{aligned}$$

Thus, under the assumption (10) and the assumption $E \left\{ \bigl| D_k(x, u, \mathbf{w}_k) \bigr|^2 \right\} > 0$, the condition (9) holds and guarantees that by directly obtaining cost difference samples rather than differencing cost samples, the error variance decreases.

3. APPROXIMATING COST-TO-GO DIFFERENCES

A common approach to compute an approximate rollout policy $\overline{\pi}$ consists of two steps:

(1) In a preliminary phase, approximations to the cost-to-go functions J_k, denoted $\tilde{J}_k$, are computed, possibly using simulation and a least squares fit from a parametrized class of functions. The techniques of NDP/RL such as temporal difference methods and others can be used for the training of such an approximation (see [BeT96] for a detailed account).

(2) Given $\tilde{J}_k$ and a state x at time k, the approximate Q-factor

$$\tilde{Q}_k(x,u) = E\big\{g(x,u,w) + \tilde{J}_{k+1}\big(f(x,u,w)\big)\big\}$$

is computed for all $u \in U(x)$, and the (approximate) rollout control $\overline{\mu}_k(x)$ is obtained by the minimization

$$\overline{\mu}_k(x) = \arg \min_{u \in U(x)} \tilde{Q}_k(x,u).$$

Unfortunately, this method also suffers from the error magnification inherent in the Q-factor differencing operation. This motivates an alternative approach, called *differential training*, that is based on cost-to-go difference approximation. We observe that to compute the rollout control $\overline{\mu}_k(x)$, it is sufficient to have the differences of costs-to-go

$$J_{k+1}\big(f(x,u,w)\big) - J_{k+1}\big(f(x,\hat{u},w)\big).$$

Thus, we consider approximating this difference using a function approximator, which given any two states s and $\hat{s}$, gives an approximation $\tilde{G}_{k+1}(s,\hat{s})$ of $J_{k+1}(s) - J_{k+1}(\hat{s})$. The rollout control can then be approximated by

$$\overline{\mu}_k(x) = \arg \min_{u \in U(x)} E\big\{g(x,u,w) - g(x,\mu(x),w) + \alpha\tilde{G}_{k+1}\big(f(x,u,w), f(x,\mu_k(x),w)\big)\big\}. \tag{11}$$

An important point here is that the training of an approximation architecture to obtain $\tilde{G}_{k+1}$ can be done using any of the standard NDP/RL methods [e.g., TD(λ), approximate and optimistic policy iteration or λ-policy iteration, etc.]. To see this, denote for all k

$$G_k(x,\hat{x}) = J_k(x) - J_k(\hat{x}).$$

By subtracting the DP equations corresponding to π, and to x and $\hat{x}$,

$$J_k(x) = E\big\{g\big(x,\mu_k(x),w\big) + J_{k+1}\big(f(x,\mu_k(x),w)\big)\big\},$$

$$J_k(\hat{x}) = E\big\{g\big(\hat{x},\mu_k(\hat{x}),w\big) + J_{k+1}\big(f(\hat{x},\mu_k(\hat{x}),w)\big)\big\},$$

we obtain, for all $(x, \hat{x})$ and k,

$$G_k(x, \hat{x}) = E\big\{g\big(x, \mu_k(x), w\big) - g\big(\hat{x}, \mu_k(\hat{x}), w\big) + G_{k+1}\big(f(x, \mu_k(x), w), f(\hat{x}, \mu_k(\hat{x}), w)\big)\big\}.$$

Therefore, G_k can be viewed as the cost-to-go function for a problem involving a fixed policy, the state $(x, \hat{x})$, the cost per stage

$$g\big(x, \mu_k(x), w\big) - g\big(\hat{x}, \mu_k(\hat{x}), w\big), \tag{12}$$

and the state transition equation

$$(x_{k+1}, \hat{x}_{k+1}) = \big(f(x_k, \mu_k(x_k), w_k), f(\hat{x}_k, \mu_k(\hat{x}_k), w_k)\big). \tag{13}$$

Thus, it can be seen that *any of the standard methods that can be used to train architectures that approximate J_k, can also be used for training architectures that approximate G_k.* For example, one may use simulation-based methods, such as TD(λ), that generate trajectories consisting of pairs $(x, \hat{x})$ according to Eq. (13). Note that a single random sequence $\{w_0, \ldots, w_{N-1}\}$ may be used to simultaneously generate samples of $G_k(x, \hat{x})$ for several triples $(x, \hat{x}, k)$. This is similar in spirit to the ideas of Perturbation Analysis (see e.g., Cassandras [Cas93], Glasserman [Gla91], or Ho and Cao [HoC91]).

The approach just described is well suited to a policy iteration context, where subsequent to the relative cost-to-go approximation corresponding to the base policy π, we will be interested in constructing a relative cost-to-go approximation corresponding to the rollout policy $\overline{\pi}$. For this construction, simulation will be used, during which it is essential that the next policy $\overline{\pi}$ can be fairly easily generated. Using Eq. (11) is quite convenient for this purpose.

A special case of interest arises when a linear, feature-based architecture is used for the approximator $\tilde{G}_k$. In particular, let F_k be a feature extraction mapping that associates a feature vector $F_k(s)$ with state s and time k, and let $\tilde{G}_k$ be of the form

$$\tilde{G}_k(s, \hat{s}) = r(k)'\big(F_k(s) - F_k(\hat{s})\big),$$

where $r(k)$ is a tunable weight vector of the same dimension as $F_k(s)$ and prime denotes transposition. The rollout policy is generated by

$$\overline{\mu}_k(x) = \arg \min_{u \in U(x)} E\big\{g(x, u, w) + r(k)'F_k\big(f(x, u, w)\big)\big\},$$

which corresponds to using $r(k)'F_k(s)$ (plus an unknown inconsequential constant) as an approximator to $J_k(s)$. Thus, in this approach, *we essentially use a linear, feature-based architecture to* *approximate J_k, but we train this architecture using the differential system (13) and the differential cost per stage (12).*

The preceding discussion also applies to infinite horizon problems. The analog of Eq. (11) is

$$\overline{\mu}(x) = \arg \min_{u \in U(x)} E\big\{g(x,u,w) - g(x,\mu(x),w) + \alpha \tilde{G}\big(f(x,u,w), f(x,\mu(x),w)\big)\big\},$$

and $\tilde{G}$ is an approximation to the cost-to-go difference, defined for any pair of states $(x,\hat{x})$ by

$$G(x,\hat{x}) = J(x) - J(\hat{x}).$$

By subtracting the two Bellman's equations corresponding to μ, and to x and $\hat{x}$,

$$J(x) = E\big\{g\big(x,\mu(x),w\big) + \alpha J\big(f(x,\mu(x),w)\big)\big\},$$
$$J(\hat{x}) = E\big\{g\big(\hat{x},\mu(\hat{x}),w\big) + \alpha J\big(f(\hat{x},\mu(\hat{x}),w)\big)\big\},$$

we obtain

$$G(x,\hat{x}) = E\big\{g\big(x,\mu(x),w\big) - g\big(\hat{x},\mu(\hat{x}),w\big) + \alpha G\big(f(x,\mu(x),w), f(\hat{x},\mu(\hat{x}),w)\big)\big\}, \quad \forall\ (x,\hat{x}).$$

Therefore, G can be viewed as the cost-to-go function for a problem involving a fixed policy, the state $(x,\hat{x})$, the cost per stage

$$g\big(x,\mu(x),w\big) - g\big(\hat{x},\mu(\hat{x}),w\big), \tag{14}$$

and the state transition equation

$$(x_{k+1},\hat{x}_{k+1}) = \big(f(x_k,\mu(x_k),w_k), f(\hat{x}_k,\mu(\hat{x}_k),w_k)\big). \tag{15}$$

The simulation-based training of $\tilde{G}$ can be done by using the differential cost (14) and the differential system (15).

4. CONCLUSIONS

The main theme of this paper, namely that one should try to approximate Q-factor and cost-to-go differences rather than Q-factors has been emphasized by a number of authors. In particular, it has been discussed by Werbos [Wer92a], [Wer92b] in several contexts, including continuous-time deterministic optimal control, and by Baird, Harmon, and Klopf [Bai93], [HBK94] in the context of advantage updating. The differential training method proposed in this paper, is new to the author's knowledge, and has the significant advantage that it requires straightforward adaptations of well-known and tested methods such as TD(λ) and λ-policy iteration. We only have preliminary computational experience with the differential training method, which is however, impressively positive, and has been successful in problems where the standard training methods have failed. A somewhat restrictive assumption in our discussion is the independence of the disturbances w_k. In many problems of interest, w_k depends on preceding disturbances via the current state x_k and/or the control u_k. On the other hand, one can often introduce approximations in the approximation model to eliminate such dependencies.

REFERENCES

[BBS95] Barto, A. G., Bradtke, S. J., and Singh, S. P., 1995. "Learning to Act Using Real-Time Dynamic Programming," Artificial Intelligence, Vol. 72, pp. 81-138.

[BTW97] Bertsekas, D. P., Tsitsiklis, J. N., and Wu, C., 1997. "Rollout Algorithms for Combinatorial Optimization," Report LIDS-P-2386, Lab. for Information and Decision Systems, Mass. Institute of Technology, Cambridge, MA; to appear in Heuristics.

[Bai93] Baird, L. C., 1993. "Advantage Updating," Report WL-TR-93-1146, Wright Patterson AFB, OH.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[Cas93] Cassandras, C. G., 1993. "Discrete-Event Systems: Modeling and Performance Analysis," Aksen Associates, Boston, MA.

[Gla91] Glasserman, P., 1991. "Gradient Estimation via Perturbation Analysis," Kluwer, Boston, MA.

[HBK94] Harmon, M. E., Baird, L. C., and Klopf, A. H., 1994. "Advantage Updating Applied to a Differential Game," unpublished report, presented at the 1994 Neural Information Processing Systems Conference, Denver, CO.

[HoC91] Ho, Y. C., and Cao, X., 1991. "Perturbation Analysis of Discrete-Event Systems," Kluwer, Boston, MA.

[TeG96] Tesauro, G., and Galperin, G. R., 1996. "On-Line Policy Improvement Using Monte Carlo Search," unpublished report, presented at the 1996 Neural Information Processing Systems Conference, Denver, CO.

[TsV96] Tsitsiklis, J. N., and Van Roy, B., 1996. "An Analysis of Temporal-Difference Learning with Function Approximation," Lab. for Info. and Decision Systems Report LIDS-P-2322, Massachusetts Institute of Technology, Cambridge, MA.

[Tsi94] Tsitsiklis, J. N., 1994. "Asynchronous Stochastic Approximation and Q-Learning," Machine Learning, Vol. 16, pp. 185-202.

[Wer92a] Werbös, P. J, 1992. "Approximate Dynamic Programming for Real-Time Control and Neural Modeling," D. A. White, D. A. Sofge, (eds.), Handbook of Intelligent Control, Van Nostrand, N. Y.

[Wer92b] Werbös, P. J, 1992. "Neurocontrol and Supervised Learning: an Overview and Valuation," D. A. White, D. A. Sofge, (eds.), Handbook of Intelligent Control, Van Nostrand, N. Y.