IDIAP RESEARCH REPORT

Gradient Estimates of Return

Christos Dimitrakakis [a] Samy Bengio [b]

IDIAP–RR 05-29

March 2006

[a] IDIAP, CP952, 1920 Martigny, Switzerland, dimitrak@idiap.ch
[b] IDIAP, CP952, 1920 Martigny, Switzerland, bengio@idiap.ch

IDIAP Research Institute **www.idiap.ch**
Rue du Simplon 4 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 05-29

Gradient Estimates of Return

Christos Dimitrakakis Samy Bengio

March 2006

Abstract. The exploration-exploitation trade-off that arises when one considers simple point estimates of expected returns no longer appears when full distributions are considered. This work develops a simple gradient-based approach for mainting such distributions and investigates methods for using them to direct exploration.

1 Introduction

A large number of problems in both supervised and reinforcement learning are solved with parametric methods. In this framework we attempt to approximate a function $f^*(\cdot)$ via a parameterised function $f(\theta, \cdot)$, given samples of f^*, with parameters $\theta \in \mathbb{R}^n$. We focus on incremental optimisation methods for which an optimisation operator $\mathcal{M}(C, \theta)$, where C is an appropriately defined cost, can be defined as a stochastic process that is continuous with respect to θ. We define the sequence $\{\theta\}$ as $\theta_{t+1} = \mathcal{M}(C, \theta_t)$.

In reinforcement learning, samples of f^* are generated actively. Asymptotic convergence results exist for such methods under appropriate sampling assumptions [3]. If we maintain a distribution of θ_t (rather than a point estimate), we may be able to use it to generate samples in an optimal sense.

Simple gradient-based methods can be used for measuring the accuracy of our parameter estimates, since intuitively, when the gradient is close to zero, our parameter estimates have converged and are thus accurate. In this paper we explore ways to measure the accuracy of parameters estimated via stochastic gradient descent methods. This involves estimating the gradient vector (and possibly the Hessian) and subsequently using it as a measure of convergence. Two cases are considered: variance estimates and gradient estimates. A naive variance estimate, arising from smoothness assumptions, is given and its relation to the gradient is detailed. The relation of the gradient to convergence is outlined and finally a simple gradient estimate is given.

1.1 Variance estimates

In the general setting, for each θ_t we sample a single value M_t from $\mathcal{M}(C, \theta_t)$, where $\mathcal{M}$ is considered as a random process. In our setting we will attempt to also maintain a confidence measure for our parameters. We will attempt to do this by measuring the variance of the process at the current point θ_t.

Firstly, we assume that M_t is bounded[1] and we attempt to estimate $\hat{E}[M_t] \approx E[M_t]$.

We may further assume that $\mathcal{M}$ is Lipschitz continuous with respect to θ, (a function f satisfies a Lipschitz continuity assumption in some set S if there exists $L \in \mathbb{R}$ such that $\|\nabla f(a) - \nabla f(b)\| \leq L\|a - b\|$ for all $a, b \in S$). An alternative, though not strictly equivalent, way of expressing this continuity is to place a prior over time for the statistics of the operator. The following simple relation follows from the assumption of an exponential prior dependency (see Appendix B) for the variance of the operator $\mathcal{M}$:

$$V_{t+1} = (1 - \zeta)V_t + \zeta(\hat{E}[M_t] - \theta_{t+1})(\hat{E}[M_t] - \theta_{t+1})', \tag{1}$$

with $\zeta \in [0, 1]$, where we have but one sample of $\mathcal{M}(C, \theta_t)$ for each time t and we make use of our smoothness assumptions for estimating variances. Now we may use V_t for our estimate of the variance of $\mathcal{M}(C, \theta_t)$.

In order to get useful estimates, we need some expressions for $\hat{E}[M_t]$. We examine the two simplest cases:

1.1 Definition (Naive variance estimate) *By assuming that $\mathcal{M}$ is a zero-mean process, i.e. that $E[M_t] = \theta_t$, we have:*

$$V_{t+1} = (1 - \zeta)V_t + \zeta(\theta_t - \theta_{t+1})(\theta_t - \theta_{t+1})'. \tag{2}$$

1.2 Definition (Counting variance estimate) *By assuming $E[M_t] = \theta_{t+1}$, e.g. when $\mathcal{M}$ is a deterministic process, we have:*

$$V_{t+1} = (1 - \zeta)V_t. \tag{3}$$

[1] For stochastic gradient methods, under the condition that the partial derivative of the cost with respect to the parameters is bounded, all M_t are bounded.

The latter method is equivalent to a class of counting schemes whereby we increase our certainty about the mean of some random variable with each observation. With an appropriate choice for ζ such schemes can be adequate for some problems.

We may further add a small positive constant to the above updates such that the variance does not eventually reach zero, if it is desirable. In the case where we maintain a set of parameters which are updated separately (such as in tabular reinforcement learning methods, which are further discussed in Section 1.3.2), then it is also appropriate to maintain separate variance estimates. In the following section we discuss how such estimates are related to the convergence of the stochastic operator $\mathcal{M}$ for the case when it expresses a stochastic gradient descent step.

1.2 Relation of variance estimates to convergence

In estimation problems we wish to find the unknown parameters θ^* from observations. In iterative estimation methods we would like to the distance between the current estimate θ and the unknown parameters, so that the process can be stopped. Unfortunately, estimating $|\theta - \theta^*|$, the distance to a solution, can be as difficult as determining θ^* itself. On the other hand, it is generally not possible to determine convergence, in certain special cases it presents a manageable task. To give a simple example, when the cost surface is quadratic (i.e $C = a(\theta^* - \theta)^2$) we have $|\theta^* - \theta| = a|\nabla_\theta C|$ and the magnitude of the steps we are taking is directly related to the convergence. It is easy to show that the mean update we have defined is an approximate measure of the gradient under some conditions.

From (1), we have

$$\begin{aligned} V_{t+1} &= (1-\eta)V_t + \eta\alpha(\delta_t + e_t)'(\delta_t + e_t) \\ &= (1-\eta)^t V_1 + \sum_{k=1}^{t} (1-\eta)^{t-k} \eta\alpha(\delta_k + e_k)'(\delta_k + e_k) \\ &= (1-\eta)^t V_1 + \eta\alpha(\sum_{k=1}^{t}(1-\eta)^{t-k}\delta_k'\delta_k + \sum_{k=1}^{t}(1-\eta)^{t-k}e_k' e_k) + 2\sum_{k=1}^{t}(1-\eta)^{t-k}\delta_k' e_k) \end{aligned} \tag{4}$$

where e_k is a noise process such as the stochastic gradient error term. For the case when $\eta = 1/t$ we have, with better approximation as $t \to \infty$, and if $\delta_k = C(\theta)$ for all k (i.e. when $\alpha \to 0$)

$$\text{trace}(V) \propto \|\nabla C(\theta)\|^2 + E^2[e],$$

where the right hand term from the stochastic gradient method is proportional to the variance of the observation noise in the linear case.

1.2.1 Gradient estimates

The relation of those estimates to the gradient is of interest because of the relationship of the gradient to the distance from the minimum under certain conditions. In particular, when $\nabla^2 C(\theta)$ is positive definite, the following holds:

1.1 Lemma *Let θ^* be a local minimum of C and $\theta \in S$, with $S = \{\theta : \|\theta - \theta^*\| < \delta\}$, $\delta > 0$. If there exists $m > 0$ such that*

$$m\|z\|^2 \leq z'\nabla^2 C(\theta) z, \quad \forall\ z \in \mathbb{R}^n, \tag{5}$$

then every $\theta \in S$ satisfying $\|\nabla C(\theta)\| \leq \epsilon$ also satisfies

$$\|\theta - \theta^*\| \leq \epsilon/m, \quad C(\theta) - C(\theta^*) \leq \epsilon^2/m.$$

The proof is quite straightforward and is given in Appendix A. We may now define a simple estimate for the gradient itself.

1.3 Definition (Gradient estimate) *By using similar assumptions as in the variance estimates, we may obtain an estimate of the gradient at time t:*

$$U_{t+1} = (1 - \zeta)U_t + \zeta(\hat{E}[M_t] - \theta_{t+1}) \tag{6}$$

Here U_t is our current estimate of the gradient vector and ζ represents our belief on how fast it changes between iterations.

1.3 Action selection

Most, if not all, reinforcement learning (RL) methods can be viewed as a combination of estimation and sampling. Given a state space $\mathcal{S}$ and an action space $\mathcal{A}$, an agent selects actions $a \in \mathcal{A}$ according to a policy $\pi : \mathcal{S} \to \mathcal{A}$. The aim of reinforcement learning is described as finding a policy π^* that maximises a utility function, for which the only available information is reward samples r_t. The utility function we shall be examining is the expectation of the return given the policy.

An important subset of reinforcement learning methods is formed by value-based methods (which are the focus of [6]). These generate an evaluation for every possible action and state pair and the policy is defined in terms of this. State-action evaluations are usually noted in short-hand as $Q(s, a) = \hat{E}[R_t|s_t = s, a_t = a, \pi]$, i.e. the expected cost/return if we take action a at state s while following policy π. Value function updates typically employ temporal-difference methods, whereby parameters are adjusted in the direction of the temporal-difference error, which has the form $\delta = r_t + \gamma\hat{E}[R_{t+1}|s_{t+1}, a_t, \pi] - Q(s, a)$. In some cases parameters are adjusted according to an importance weight, which usually takes the form of an *eligibility trace* e_i, defined for each parameter θ_i.

1.3.1 Application of variance estimates to action values

These variance estimates can be applied with relative ease to action value reinforcement learning using either a tabular or an approximation architecture. The naive variance estimate (2) is particularly interesting because, for the tabular case, its use results in algorithm that is similar to [5]. For this reason we shall concentrate on this particular estimate, but we will also be contrasting it to a gradient-related estimate.

In the following short sections we consider the application of such an estimate to reinforcement learning; firstly in the tabular and secondly in the function approximation case. Lastly, we describe action selection mechanisms, using the developed variance estimates, that can be applied to either case.

1.3.2 Tabular action value methods

The tabular reinforcement learning case can be obtained by defining a θ for each state-action pair Q, so that we maintain separate variance estimates for each one. Then we consider that at each time step the operator sample M_t can be defined as $M_t \equiv Q_{t+1}(s, a) = Q_t(s, a) + \alpha(r_t + \hat{E}[R_{t+1}] - Q_t(s, a))$. By substituting this into (2), we obtain

$$V_{t+1} = (1 - \zeta)V_t + \zeta\delta\delta', \tag{7}$$

where $\delta = Q_{t+1} - Q_t$ is the temporal-difference error vector (scaled by the step-size constant α). For the standard tabular case, all elements of δ will be 0 apart from the element corresponding to the action a, which is the one to be updated and the covariance matrix $\delta\delta'$ will have a single non-zero diagonal element.

By re-arranging the terms of (7) we arrive at

$$V_{t+1} - V_t = \zeta(\delta\delta' - V_t) \tag{8}$$

which can be written in expanded form as

$$V_{t+1}(s, a) - V_t(s, a) = \zeta(\delta(s, a) - V_t(s, a)). \tag{9}$$

In the following we briefly describe how eligibility traces can be integrated in our framework.

1.3.3 Eligibility traces and variance estimates

Let us assume that the return R_t is given by a probability distribution of the form $p(R_t|s_t, a_t, \pi)$. Clearly, we may estimate $E[R_t|s_t, a_t, \pi]$ by averaging the returns while following policy π. However, we can assume that the distribution of R_t depends upon the distribution of R_{t+1}. We assume an exponential distribution for this prior dependency and thus we have $p(R_{t+1}|s_{t+1}, a_{t+1}, \pi) = \lambda p(R_{t+1}|s_t, a_t, \pi) + (1-\lambda)\mathcal{W}$, where $\mathcal{W}$ is the distribution of some unknown process, while $\lambda \in [0, 1]$ represents our prior belief in the dependency between values at different times.

The relation to eligibility traces is clear. We assume that an exponential prior in time governs the probability distribution of R_t. Thus, we can perform importance sampling on our parameters through the use of this prior: in other words each new sample should influence each parameter according to its importance weight.

In RL methods employing eligibility traces, the update δ is applied to all the evaluations of all state-action pairs (s, a) proportionally to the eligibility trace $e(s, a)$. By viewing eligibility traces as importance weights we can integrate them easily with our variance estimates. This results in the following update for each parameter's estimate.

$$V_{t+1}(s, a) = (1 - \zeta e(s, a))V_t(s, a) + \zeta e(s, a)\delta\delta', \tag{10}$$

or in compact form

$$V_{t+1} = (I - \zeta e)V_t + \zeta e\delta\delta', \tag{11}$$

where I is the identity matrix.

1.3.4 Function approximation methods

We consider approaches where the value function is approximated with a parametrised function $Q_\theta : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$.

Gradient methods are a commonly used method for adapting the parameters θ. Given $\frac{\partial Q}{\partial \theta}\frac{\partial C}{\partial Q} \equiv \nabla_\theta Q \nabla_Q C$, we consider an update of the form $M_t = \theta_t + d_t$ for our parameters, where d_t is the gradient descent update. Then we simply apply (7) for this case and we obtain a covariance matrix for the parameters.

1.4 A unified view of action selection

We are proposing two methods for using our variance estimates. The first is an intuitive action selection mechanism for the case of linear approximations (including the tabular case), since the variance of Q depends linearly on those, which allows us to analytically define a distribution over actions. An alternative, and in our view more interesting, approach is to sample directly from the distribution of parameters. The two methods are described in more detail in the following.

1.4.1 Sampling actions from action distributions

Consider the probability $P(a^* = a|s)$ of action a being the optimal action for some state s. We need to obtain the posterior distribution of this for all actions, given the distribution of Q and the state, [2] denoted $P(a^* = a|Q, s)$. The Bayes rule gives

$$P(a^* = a|Q, s) = \frac{p(Q|a^* = a, s)P(a^* = a|s)}{\sum_{b\in\mathcal{A}} p(Q|a^* = b, s)P(a^* = b|s)}, \tag{12}$$

[2]In our model Q is no longer a single value but a distribution characterised by the variance V. In this section we make no distinction between our estimate of the mean and the actual distribution in order to minimise symbol use.

where we have made use of the fact that $\sum_{b \in \mathcal{A}} P(a^* = b|s) = 1$. Now we must assume a distribution family for $p(Q|a^*, s)$. We consider the Boltzmann distribution which can be written as

$$p(E|i) = \exp(-E_i/K\tau)$$

and has a physical interpretation of the distribution of the energies E of particles in a given state i depending on the temperature τ and a constant factor K. We will be using this in the following to define a soft-max method for selection actions:

1.4 Definition (weighted-softmax) *Select actions a according to probabilities:*

$$P(a^* = a|Q, s) = \frac{\exp(Q(s,a)/\sqrt{v_{s,a}})}{\sum_b \exp(Q(s,b)/\sqrt{v_{s,b}})} \ . \tag{13}$$

For the tabular case, $v_{s,a}$ at time t is simply $V_t(s,a)$. For the linear case, in which $Q(s,a) = \sum_i w_{i,a} s_i$, where w are components of a weight matrix and s_i is the i-th component of the state vector, the variance is simply $v_{s,a} = \sum_i w_{i,a} V_t(i,a)$ where $V_t(i,a)$ is the variance of the weight $w_{i,a}$. Of course we could also consider a full covariance matrix.

Another possibility is to consider an approximation to the action distribution, given the distribution of parameters, which is certainly more elegant. However, it is probably simpler to sample directly from the distribution of parameters and this is the approach we outline below.

1.4.2 Sampling actions via parameter sampling

The second method applies to the more general function approximation case, as well as the tabular case. Here we have to choose a distribution for our parameters; and then we sample from it in order to generate actions, rather than postulating a distribution over actions and sampling from that. This is because in the general case it is difficult to determine the distribution over actions from that of the parameters, while in any case it is sufficient to sample from the parameter distribution directly.

1.5 Definition (Sampling-Greedy) *In this method, action sampling arises from sampling in the parameter space. At each time, we select action a^* such that*

$$a^* = \arg\max_a Q(s, a, \Theta), \tag{14}$$

where $\Theta = \mathcal{N}(\theta, V_t)$ is a sample randomly drawn from a normal distribution with mean θ and variance V_t.

This will select action a^* with probability $\prod_{a \in \mathcal{A}} P(Q(s,a^*) > Q(s,a)|\Theta)$, if we consider Θ as our belief distribution. This means that it will select each action with probability proportional to the probability of it being the best action.

1.5 Alternative approaches

There have been previous applications of such methods to the problem of action selection in reinforcement learning. In [2], the authors take a Bayesian approach for estimating parameter distributions. They also consider two different action selection schemes: (a)Q-value sampling, in which the probability that the return of one action is larger than those of others is approximately calculated, which is similar to sampling-greedy selection mechanism presented here (Definition 1.5) and (b)Myopic-VPI selection, in which the value to be gained by exploring actions is estimated. This method differs from ours particularly in the sense that a Bayesian framework is employed. However the authors concede that the application of a full Bayesian framework is intractable in reinforcement learning and make do with some approximations. The authors specify a great many optional features in their framework, for a particular subset of which our methods only differ in the choice of distribution. More specifically, the moment updating scheme and Q-value sampling that is described therein is quite similar to choosing the sampling-greedy scheme together with the naive variance update in our model.

The second approach that we are currently aware of is the Reliability Index method, described in [5]. This method has substantial similarity to our own for tabular action value methods using the naive variance update (2), so we pause for a moment to ponder the differences. In this method, actions are sampled according to a Boltzmann distribution:

1.6 Definition (Reliability Index)

$$p(a|Q,s) = \frac{\exp(\eta Q(s,a)/\sqrt{v_s})}{\sum_{b \in A} \exp(\eta Q(s,b)/\sqrt{v_s})} \tag{15}$$

where $v_s > 0$ is defined $\forall\ s \in \mathcal{S}$ and is a variance estimate *for each one of our Q estimates and η is a free parameter.*

In that method, the variance update assumes the form $V_{t+1}(s) - V_t(s) = \zeta(\delta(s)\delta(s)' + \gamma V_t(s') - V_t(s))$, with a common V for all actions, or of a type $V_{t+1}(s,a) - V_t(s,a) = \zeta(\delta(s)\delta(s)' + \gamma V_t(s',a) - V_t(s,a))$. This is then averaged over all states to obtain $V_t(s) = \frac{1}{|\mathcal{A}|}\sum_a V_t(s,a)$. In either case, actions are selected according to (15). It is perhaps important to note that a temporal-difference type of update is used (since $V_t(s')$ is the estimated variance of next state's evaluation). The authors postulate that this represents the dependency between the reliability of one state and the ones close to it. In our view, parameter variances are directly related to parameter updates and γ is related to the utility function rather than to assumptions about parameter dependencies. However, a model for setting priors about parameter dependencies is offered by the exponential prior, commonly used in eligibility traces. This is the method we have employed, as explained in Section 1.3.3.

Because of the close relation between this method and ours, our results can be viewed as, firstly, a partial justification of the RI method and secondly, the generalisation of the concept to arbitrary action-value methods.

2 Experiments

In this section we discuss results on a few simple discrete state tasks: a n-armed bandit problem, graph walking and pole balancing. In the n-armed **bandit task**, the environment is composed of a single state and n actions, with each action a having a reward $r_a \in \{0,1\}$ governed by a Bernoulli distribution. In this environment, all actions apart of one have a mean reward of 0.2, while the optimal one has a mean of 0.3. The **graph task** consists of m nodes and n actions. Actions result in transitions from one node to another deterministically. The first action always results in a transition from one node to the next, forming a ring. Thus, starting from some node i it is possible to visit all nodes in m moves by simply choosing action 1. Other actions' transitions are determined randomly at the beginning of each experimental round. The reward distribution for each state is Bernoulli, with mean in $[0,b]$ selected randomly at the beginning of each round from the bounded distribution

$$P(X < x) = 1/b^2 \int_0^b \frac{b - x/t}{b} dt,$$

where b was set to 0.5. For the **pole balancing task** we used a discrete simulation with a semi-elastic pole with a ball at one end and a cart sliding on a surface. The state consisted of the angular velocity and angle of pole, and speed and position of the cart. These were evenly discretised to form a state space of size 256. There were 6 possible actions, which resulted in a lateral force being applied to the cart. The reward was 0 for all time, apart from failure, when it was -1 and the apparatus was reset to a random position.

We have compared our proposed method, together with one of the three variance estimates (counting, naive and velocity), and one of the two action selection methods (sampling and weighted-softmax), against standard action selection techniques and the reliability index method. For tasks with state, we also explored the eligibility trace update option for variance estimates, as outlined previously. For all methods the learning rate was set to 0.01. For tasks with state, λ was set to 0.7 and γ was set to 0.9 and Q-learning was employed. Each method has a free parameter, such as the temperature, or ζ, which we varied in the range $[10^{-4}, 1]$. To examine the performance of the methods, we use the total reward achieved after a fixed number of time steps, averaged over experimental runs. This measure enables us to summarise performance in terms of quality of final solution and speed of convergence simultaneously.

2.1 Results

In the **bandit task**, the number of arms was varied from 2 to 200. For a small number of arms the various methods performed equally well. For increasing numbers of arms, the standard softmax and ϵ-greedy methods failed to reach a satisfying solution, while the sampling method that we employed tended to perform slightly better than other methods. The weighted-softmax was performing generally worse than the other adaptive methods. In this task we did not observe a large difference between the naive variance, the velocity and the simple counting estimate. Note that when using naive variance updates, the only difference between our method and RI lies in the action selection mechanism. It would thus appear to be strange that the weighted-softmax is performing worse than RI. However, this is probably due to the fact that, if all rewards are positive, actions with high variances are penalised by this method and it frequently halts quickly at a sub-optimal solution.

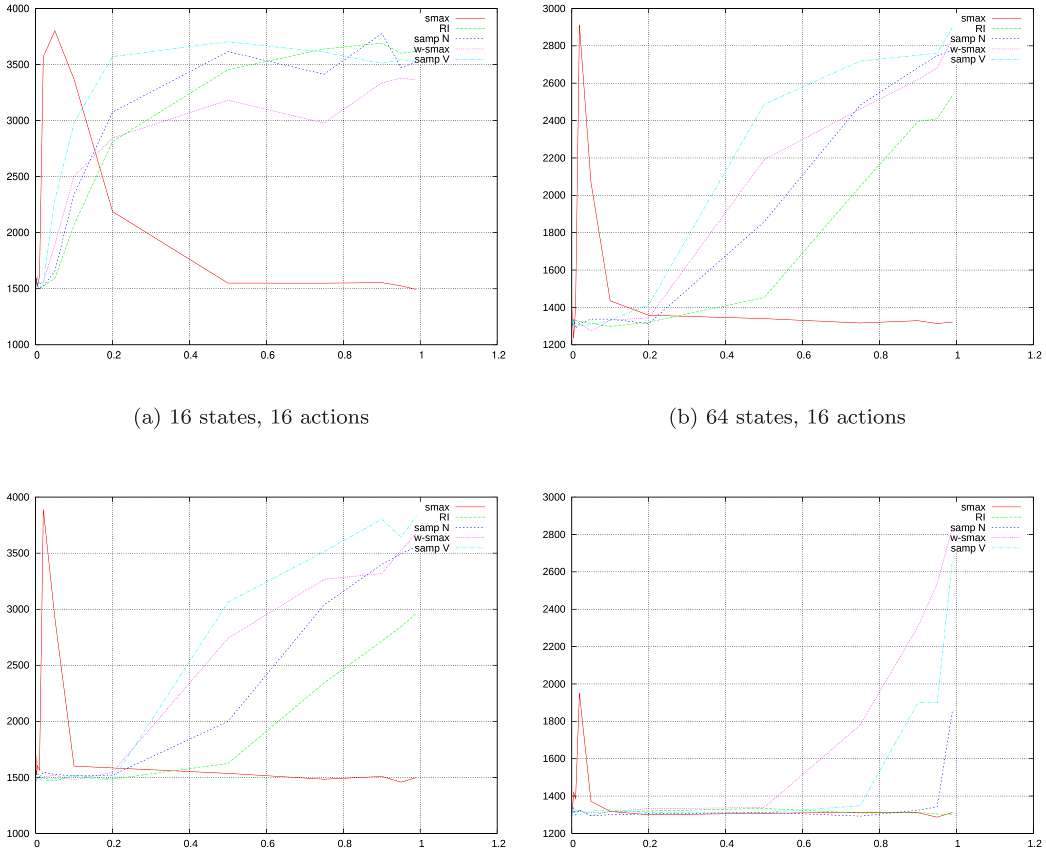
For the **graph task**, we varied the number of states in $\{4, 16, 64\}$ and the number of actions in $\{4, 16, 64\}$. While in general all methods' performance degraded with an enlargement of the space, and all methods performed similarly for small spaces, we found that as the size of the space increased, the sampling method performed significantly better than other methods. The weighted softmax method appeared to be the most sensitive to the size of the space, behaving quite badly in small spaces and much better than anything else in the 64×64 space. We also found that the velocity estimate offered some improvement, especially with regard to sensitivity to ζ, while the naive and counting estimates had an essentially identical performance. The standard softmax method achieved as good a solution as the best methods, but only for a very limited range of values for τ, while the ϵ-greedy method performed the worst.

Figure 1 summarises those results. Those indicate that for smaller sizes of the state-action space, the standard softmax is much less sensitive to the selection of the temperature parameter. Larger spaces, however, put it at some disadvantage. For the largest case examined here, the adaptive sampling methods proposed perform significantly better. All such methods tend to perform better when $\zeta \to 1$ and this effect most pronounced for the largest spaces. A tentative hypothesis for an explanation of this behaviour could be given by considering the interaction of two factors: Firstly, the determinism of the environment transitions, and secondly the fact that the number of iterations used is barely sufficient for trying out all possible state-action pairs once in the largest environment. In such an environment the most obvious approach would be to never try a seemingly inferior more than once - if the number of states is large enough, soon a good but sub-optimal solution would be found.

We used the **pole balancing task** to explore what occurs in situations when initial estimates of return are optimistic. In this and other similar tasks, a greedy policy outperformed everything else by a large margin. This is to be expected, since the nature of the reward used is sufficient for exploration to take place.

2.2 Discussion

To summarise, the adaptive exploration methods discussed herein seem to be useful in some settings. The optimal value for ζ appears to be similar across action selection mechanisms and variance estimates. It is interesting to note that using the sample velocity estimate for updates, which disregards



(a) 16 states, 16 actions

(b) 64 states, 16 actions

(c) 16 states, 64 actions

(d) 64 states, 64 actions

Figure 1: Total reward in the graph task after 10,000 iterations averaged over 100 experiments. Results are shown for softmax (smax), reliability index (RI), sampling-greedy with Naive (samp N) and Velocity (samp V) variance estimates and weighted-softmax (w-smax). The x-axis is the smoothing parameter ζ, or for softmax, the temperature τ.

the variance of the return, results in optimal behaviour over a larger set of values for ζ than either the sample variance or counting estimate.

We have also briefly explored the response of the various methods to scaling of the variance estimate as follows: The method, as described so far, used the gradient direction scaled by the learning rate to compute the variance update. However one could remove this dependency on the learning rate. Removing it had a strongly detrimental effect on the weighted-softmax and RI action selection methods, while the sampling-greedy method was unaffected.

Finally, we would like to mention that we observed no experimental differences when using an eligibility-type update for the variance estimates, though this could have very well not been so for other tasks.

2.3 Conclusion

In this section, we proposed a set of simple techniques for estimating parameter distributions, which can be applied to the development of action selection mechanisms. In preliminary experiments it was found that the use of the smoothed gradient estimate is particularly efficient in some tasks. Of interest is the fact that the naive variance estimates that are outlined are a generalisation of simple counting schemes and the scheme used in the prioritised sweeping algorithm [4] and that used in the RI method [5]. The connection between those estimates, the gradient, and its relation to convergence offers some justification to the previously ad hoc use of such techniques. Particularly for prioritised sweeping, the use of such an update might be advantageous for the case of stochastic rewards.

In preliminary experiments we have used a naive sampling method for action selection, wherein the actions are selected proportionally to the probability of their being the best action. Future work would include investigating the use of explicit estimates for the value of exploration, which is one of the approaches outlined in [2]. There are also some interesting theoretical questions, such as the relationship of our model, and its possible application to policy-gradient methods [see for example 1]. Such methods also maintain an explicit estimate of the gradient and perform sampling in the policy space (through sampling from the joint distribution of parameters) similarly to the sampling-greedy approach.

A Distance Bound

A.1 Lemma (Distance bound) *Let θ^* be a local minimum of C and $\theta \in S$, with $S = \{\theta : \|\theta - \theta^*\| < \delta\}$, $\delta > 0$. If there exists $m > 0$ such that*

$$m\|z\|^2 \leq z'\nabla^2 C(\theta)z, \quad \forall\ z \in \mathbb{R}^n, \tag{16}$$

then, for all $\theta \in S$,

$$\|\theta - \theta^*\| \leq \|\nabla C(\theta)\|/m, \quad C(\theta) - C(\theta^*) \leq \nabla\|C(\theta)\|^2/m.$$

For any twice continously differentiable function f, it holds that:

$$\nabla f(y) = \nabla f(x) + \int_0^1 \nabla^2 f\big(x + t(y-x)\big)(y-x)dt.$$

We apply this to C and note that $\nabla C(\theta^*) = 0$ to obtain:

$$\begin{aligned} \nabla C(\theta) &= \int_0^1 \nabla^2 C\big(\theta^* + t(\theta - \theta^*)\big)(\theta - \theta^*)dt \\ (\theta - \theta^*)'\nabla C(\theta) &= \int_0^1 (\theta - \theta^*)'\nabla^2 C\big(\theta^* + t(\theta - \theta^*)\big)(\theta - \theta^*)dt. \end{aligned}$$

From (16), we have:

$$\begin{aligned} (\theta - \theta^*)'\nabla C(\theta) &\geq m\|\theta - \theta^*\|^2 \\ \|\theta - \theta^*\|\|\nabla C(\theta)\| &\geq m\|\theta - \theta^*\|^2 \\ \|\theta - \theta^*\| &\leq \|\nabla C(\theta)\|/m, \end{aligned}$$

which concludes the first part of the proof.

The second statement can be proven by using the following second order expansion that holds for every function f that is twice continuously differentiable over an open sphere f centred at x, and with $y : x + y \in S$:

$$f(x+y) = f(x) + y'\nabla f(x) + \frac{1}{2}y'\nabla^2 f(x)y + o(\|y\|^2) \tag{17}$$

from which it follows that:

$$f(y) - f(x) = f(x + (y - x)) - f(x) = (y - x)'\nabla f(x) + \frac{1}{2}(y - x)'\nabla^2 f(x)(y - x) + o(\|y - x\|^2) \quad (18)$$

We also need the fact that

$$\min_{y \in \mathbb{R}^n} \left\{(y - x)'\nabla f(x) + m\|y - x\|^2/2\right\} = -\frac{1}{2m}\|\nabla f(x)\|^2. \quad (19)$$

(This can be proven by the fact that at the minimum, the derivative of the argument of the minimum operator will have a derivative of 0, resulting in $y^* = \frac{-\nabla f(x)}{m} + x$. A substitution completes the proof.)

From (16) and (18), we have:

$$\begin{aligned} f(x + (y - x)) - f(x) &= (y - x)'\nabla f(x) + \frac{1}{2}(y - x)'\nabla^2 f(x)(y - x) + o(\|y - x\|^2) \\ &\geq (y - x)'\nabla f(x) + \frac{m}{2}\|y - x\|^2 \end{aligned}$$

We can then replace the right hand side with its minimum, as given by (19), which gives:

$$f(x + (y - x)) - f(x) \geq -\frac{1}{m}\|\nabla f(x)\|^2.$$

We now replace

$$C(\theta) - C(\theta^*) \leq \frac{1}{2m}\|\nabla C(\theta)\|^2,$$

which concludes the second part of the proof.

We further note that if $\|C(\theta)\| \leq \epsilon$ then the following inequalities also hold:

$$\|\theta - \theta^*\| \leq \epsilon/m, \quad C(\theta) - C(\theta^*) \leq \epsilon^2/m.$$

B Exponential-family Priors in Time

Let us express the dependency of two random variables in time v_t and v_{t+k} can be via some prior ξ. In some cases the expectation of v_{t+k} given the prior and some observations can be written in linear form. Two such cases are outlined in the following.

B.1 Markov Process View

Let's assume this prior dependency for a Markov process

$$P(v_t|v_{t-1}) \quad (20)$$

We introduce another random variable, X:

$$P(v_t|X, v_{t-1})P(X|v_{t-1}) = P(X|v_t, v_{t-1})P(v_t|v_{t-1}) \quad (21)$$

and by assuming Markov property $P(X|v_t, v_{t-1}) = P(X|v_t)$, we have

$$P(v_t|X, v_{t-1}) = \frac{1}{Z}P(X|v_t)P(v_t|v_{t-1}), \quad (22)$$

where $Z > 0$ is some normalisation constant. Let's use an exponential prior of the form

$$P(v_t|v_{t-1}) \propto e^{-a\|v_t - v_{t-1}\|^2} \quad (23)$$

with $a > 0$, and a Gaussian distribution $P(X|v_t) \propto e^{-\|x_i - v_t\|^2}$ to obtain:

$$P(v_t|X, v_{t-1}) \propto e^{-\|v_t - X\|^2} e^{-a\|v_t - v_{t-1}\|^2}. \tag{24}$$

The MAP solution requires the minimisation of

$$f(v_t) = \|v_t - x_i\|^2 + a\|v_t - v_{t-1}\|^2 \tag{25}$$

so we have

$$\begin{aligned} \nabla f(v_t) &\propto (v_t - x_i) + a(v_t - v_{t-1}) = 0 \\ v_t &= \frac{x_i + a v_{t-1}}{1 + a} = (1 - \lambda)x_i + \lambda v_{t-1} \end{aligned} \tag{26}$$

where $\lambda = \frac{a}{1+a}$.

B.2 Convolutional View

For continuous time, the same result can be derived by examining the convolution of two densities, an observed density f and a prior density g and we wish to infer a conditional density $p(f|g)$.

$$p(f|g) = \int_{-\infty}^{0} f(s + a - t) g(-t) dt \qquad a > 0 \tag{27}$$

$$= \int_{-\infty}^{a} f(s - u) g(-a - u) du \tag{28}$$

For $g(t) = e^{-b|t|}$,

$$p(f(s + a)|g) = \int_{-\infty}^{0} f(s - u) g(a - u) du + \int_{0}^{a} f(s - u) g(a - u) du \tag{29}$$

$$= e^{-a}\Big(\int_{-\infty}^{0} f(s - u) g(-u) du + \int_{0}^{a} f(s - u) g(a - u) du\Big)$$

$$= e^{-a} p(f(s)|g) + (1 - e^{-a}) f(s + a), \tag{30}$$

assuming f is constant in $(s, s + a]$.

References

[1] Jonathan Baxter and Peter L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proc. 17th International Conf. on Machine Learning*, pages 41–48. Morgan Kaufmann, San Francisco, CA, 2000.

[2] Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

[3] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

[4] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103, 1993.

[5] Yutaka Sakaguchi and Mitsuo Takano. Reliability of internal prediction/estimation and its application. I. adaptive action selection reflecting reliability of value function. *Neural Networks*, 17(7):935–952, 2004.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.