Reinforcement Comparison

Peter Dayan
Centre for Cognitive Science &
Department of Physics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 3LW
Scotland

Abstract

Sutton [2] introduced a reinforcement comparison term into the equations governing certain stochastic learning automata, arguing that it should speed up learning, particularly for unbalanced reinforcement tasks. Williams' subsequent extensions [3] to the class of algorithms demonstrated that they were all performing approximate stochastic gradient ascent, but that, in terms of expectations, the comparison term has no first order effect.

This paper analyses the second order contribution, and uses the criterion that its modulus should be minimised to determine an optimal value for the comparison term. This value turns out to be different from the one Sutton used, and simulations suggest at its efficacy.

1 INTRODUCTION

Sutton [2] introduced the notion of reinforcement prediction as a way of speeding up the learning of a class of stochastic learning automata. Most previous methods made assumptions about the independence of the learning of the automata from all aspects of their reinforcement history that were not 'compiled' into their current action probabilities. Sutton reasoned that comparing the current reinforcement with some function of its frequency of delivery in the past might be helpful for determining whether or not their actions were making things worse or better. He expected particular utility for such comparisons in the difficult cases in which reinforcement delivery is unbalanced - for instance when all actions tend to be rewarded or punished.

Williams [3] analysed a related set of algorithms, which includes Sutton's, and demonstrated that they were all performing on-line, stochastic gradient ascent in the expected amount of reinforcement. This is reassuring, since it implies that the algorithms are moving in the correct direction, statistically at least. The surprising part of his analysis was that, for the particular case Sutton considered, the comparison term may be eliminated from the analysis at an early stage. The result on stochastic gradient ascent is unaffected by its value. Sutton's simulations, however, demonstrated that different comparison terms perform very differently.

Williams essentially looked at the first order term in the Taylor expansion of the function that relates the expected reinforcement to the weights determining the probability of performing the actions. Although the comparison term vanishes from this, it would not be expected to vanish from the second and higher order terms. Second order analysis should reveal for it both a rôle, and, potentially, an optimal value.

2 THEORY

2.1 WILLIAMS' ANALYSIS

Williams treats a very general problem. At any time, each of n units receives an input $\mathbf{x}^i \in \Re^p, 1 \leq i \leq n$ from some environment, and uses its weight vector $\mathbf{w}^i \in \Re^p$ to determine whether to fire or not; $y_i = 1$ or $y_i = 0$ respectively. Before it chooses its action, and before the environment evaluates the combined set of actions, each unit also chooses a reinforcement comparison value $b_{ij}, 1 \leq i \leq n, 1 \leq j \leq p$ for each component of each weight. The environment returns a global reinforcement value r that is stochastically related to the quality of the actions of the units, and each unit then updates its weight vector according to the reinforcement, its chosen action, and its reinforcement comparison values.

A simple example of such a reinforcement learning system is the two armed bandit problem, which we shall return to later. For this, the automaton has no inputs, but chooses, stochastically on the basis of a stored weight, to pull either the left arm ($y = 0$) of the bandit or the right arm ($y = 1$). The machine delivers reinforcement of $r = \pm 1$ with different probabilities

for the two arms, and the automaton has to learn, by changing the weight, which arm it is best to pull.

More formally, Williams proves that if:

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij}, \quad (1)$$

where,

- r is the reinforcement,
- b_{ij} are reinforcement baselines, which are conditionally independent of the actions y_i given the weights $\mathbf{W}$ and the inputs $\mathbf{x}^i$,
- α_{ij} is the learning rate parameter for w_{ij},
- $e_{ij} = \frac{\delta \ln g_i}{\delta w_{ij}}$ is the so-called eligibility of the weight w_{ij}, a measure of how influential it was in choosing the action,
- $g_i(\xi, \mathbf{w}^i, \mathbf{x}^i) = \mathcal{P}[y_i = \xi | \mathbf{w}^i, \mathbf{x}^i]$ is the probability the i^{th} unit emits action ξ given its weights $\mathbf{w}^i$ and its input $\mathbf{x}^i$.

then:

$$\mathcal{E}[\Delta w_{ij} | \mathbf{W}] = \alpha_{ij} \frac{\delta \mathcal{E}[r|\mathbf{W}]}{\delta w_{ij}}. \quad (2)$$

where $\mathbf{W}$ is the matrix of all $\mathbf{w}^i$.

Equation 2 implies that these algorithms are all performing stochastic gradient ascent in an averaged sense. The dependence on the values of the b_{ij} drops out at an early stage, since:

$$\mathcal{E}[e_{ij} | \mathbf{W}, \mathbf{x}^i] = 0.$$

However, looking at equation 1, it is apparent that changing the b_{ij} is likely to affect at very least the stability of the algorithm. Sutton [2] investigated this empirically for his algorithm, and found faster convergence across a range of problems for b_{ij} being estimators of the average amount of reinforcement received than for $b_{ij} = 0$.

2.2 THE SECOND ORDER TERM

Unfortunately, treating higher order terms at the same level of generality as Williams is not fruitful. Consider instead the simplest of the cases that Sutton takes. Here there is just one unit, weights w_i, inputs x_i, reinforcement r, and with:

$$\Delta w_i = \alpha(r - b)(y - \pi)x_i$$

where $\pi = \mathcal{E}[y|\mathbf{x}, \mathbf{w}]$. b can depend on $\mathbf{x}$ and $\mathbf{w}$, but not on the output y.

A different way of looking at Williams' result is through the Taylor expansion of $\mathcal{E}[r'|\mathbf{w}, \mathbf{x}]$, using the prime $'$ to indicate that it is the expected value of the reinforcement that will be received at the next time step:

$$\mathcal{E}[r'|\mathbf{w}, \mathbf{x}] = \mathcal{E}[r|\mathbf{w}] + \sum \mathcal{E}[\Delta w_i | \mathbf{w}, \mathbf{x}] \frac{\delta \mathcal{E}[r|\mathbf{w}]}{\delta w_i} + \frac{1}{2} \sum \mathcal{E}[\Delta w_i \Delta w_j | \mathbf{w}, \mathbf{x}] \frac{\delta^2 \mathcal{E}[r|\mathbf{w}]}{\delta w_i \delta w_j} + \ldots.$$

Williams deals with the first order term, showing that the first term in the product is proportional to the second, and that b makes no contribution whatsoever. Setting $\mathcal{F}(\mathbf{z}) = \mathcal{E}[r|\mathbf{z}]$, the second order term is:

$$\sum_{i,j} \frac{\delta^2 \mathcal{F}}{\delta w_i \delta w_j} x_i x_j \alpha^2 \times \sum_{\xi,\rho} \mathcal{P}[y = \xi | \mathbf{w}, \mathbf{x}] \mathcal{P}[r = \rho | \xi, \mathbf{x}] \times (\rho - b)^2 (\xi - \pi)^2. \quad (3)$$

Note that the inner sum does not depend on the value of i or j. Extracting the value $\hat{b}$ that minimises it, gives:

$$\hat{b} = \frac{\mathcal{E}[r(y - \pi)^2 | \mathbf{w}, \mathbf{x}]}{\mathcal{V}[y|\mathbf{w}, \mathbf{x}]}$$

However, y can only take on two values: 0 or 1. Let:

$$p = \mathcal{P}[y = 1 | \mathbf{w}, \mathbf{x}]$$
$$r_0 = \mathcal{E}[r | y = 0, \mathbf{w}, \mathbf{x}]$$
$$r_1 = \mathcal{E}[r | y = 1, \mathbf{w}, \mathbf{x}]$$

then $\pi = p$, and

$$\hat{b} = \frac{p(1-p)^2 r_1 + (1-p)p^2 r_0}{p(1-p)} = (1-p)r_1 + p r_0.$$

in which, counterintuitively, the expected reward for emitting action **1** is paired with the probability of emitting action **0**, and *vice-versa*. Williams (personal communication) derived the same expression for $\hat{b}$ on the grounds of minimising the variance of the Δw_i.[1] He ultimately considered this an inappropriate reason for choosing the value of b.

The reinforcement comparison algorithm favoured by Sutton involves teaching an extra unit to predict the future reinforcement level. He defines $s = \sum_i v_i x_i$, where $\mathbf{v}$ are the prediction weights. These are changed according to:

$$\Delta v_i = \beta(r - s)x_i.$$

This tends to make s an estimator of sorts of $\mathcal{E}[r|\mathbf{w}, \mathbf{x}]$, or b', where:

$$b' = p r_1 + (1 - p) r_0.$$

which, *a priori*, is the more natural pairing.

[1] Minimising the variance leads to the same expression since $\mathcal{E}[\Delta w_i]$ is independent of b, and x_i factors out.

2.3 CHOOSING b

Although $\hat{b}$ minimises the inner sum in the second order term of equation 3, it is not yet clear that this is appropriate. In the one dimensional case, since the reinforcement is bounded above and below, the second derivative $\delta^2\mathcal{F}/\delta w^2$ will be positive for some values of w and negative for others. This means that it is bound both to speed and hinder the learning. Setting $b = \hat{b}$ minimises this effect.

As an example, consider the first task Sutton investigated, which is a two-armed bandit problem. Here, there are two possible actions $y = 0, 1$, and:

$$\begin{array}{lll} y=0 \Rightarrow & \mathcal{P}[r=1]=0.8 & \mathcal{P}[r=-1]=0.2 \\ y=1 \Rightarrow & \mathcal{P}[r=1]=0.9 & \mathcal{P}[r=-1]=0.1 \end{array}$$

so the optimal action is $y = 1$.

Choose $\mathcal{P}[y=1|w] \equiv f(w) = 1/(1+e^{-w})$, then:

$$\begin{aligned} \mathcal{E}[r|w] &= 0.6 + 0.2f(w) \\ \frac{\delta}{\delta w}\mathcal{E}[r|w] &= 0.2f(w)(1-f(w)) \\ \frac{\delta^2}{\delta w^2}\mathcal{E}[r|w] &= 0.2f(w)(1-f(w))(1-2f(w)) \end{aligned}$$

So, with $\Delta w = \alpha(r-b)(y-\pi)$, the changes are:

y	r	$\mathcal{P}$	$\Delta w/\alpha$
0	-1	$0.2(1-f(w))$	$(1+b)f(w)$
0	1	$0.8(1-f(w))$	$-(1-b)f(w)$
1	-1	$0.1f(w)$	$-(1+b)(1-f(w))$
1	1	$0.9f(w)$	$(1-b)(1-f(w))$

Then $\mathcal{E}[\Delta w] = \alpha \times 0.2f(w)(1-f(w))$, which, as expected, is independent of b.

However, let $g(w) = \mathcal{E}[r|w] = 0.6 + 0.2f(w)$, then:

$$\begin{aligned} \mathcal{E}[\overline{r}|w] = {} & 0.2(1-f(w))g(w+\alpha f(w)(1+b)) + \\ & 0.8(1-f(w))g(w-\alpha f(w)(1-b)) + \\ & 0.1f(w)g(w-\alpha(1-f(w))(1+b)) + \\ & 0.9f(w)g(w+\alpha(1-f(w))(1-b)), \end{aligned}$$

where $\overline{r}$ is the reinforcement received after the automaton's next choice. b will not drop out of this. It is apparent from a graph of the second order term that it helps learning for $w < 0$ and hinders it for $w > 0$. Setting $b = \hat{b}$ minimises both these effects.

The same will be true in higher dimensions, in that the second order term will be alternately a hindrance and a help. Minimising its modulus should therefore increase the overall efficacy of gradient ascent, which operates perfectly on linear functions.

There are two types of imbalance that can afflict problems like the two-armed bandit:

- Imbalance in the probabilities - in which both the better and the worse action usually lead to the same value of reinforcement, the only different being in the precise frequency.
- Imbalance in the reinforcement values - in which the actual reinforcement values received are not centred around 0. This can make learning substantially more difficult by making the sign of the changes in the weights on any one occasion independent of the reinforcement received.

Reinforcement comparison only deals with the second of these types of imbalance. Williams (personal communication) has pointed out that the term $r - b_{ij}$ in the formula for the weight change, equation 1, will take both positive and negative values if the b_{ij} lie between the maximum and minimum reinforcement values. Barto [1] provides some reasons why the term $y - \pi$ in the learning rule helps mitigate the effects of the first type of imbalance.

3 RESULTS

Calculating the 'optimal' $\hat{b}$ is more difficult than calculating Sutton's $\hat{b}'$, because of the cross-pairing of the average reinforcement for action 1 with the probability of doing action 0. It is possible to develop an estimator $p^t(\mathbf{x}) = \sum v_i^t x$ with weights $\mathbf{v}$, as in Sutton's algorithm, and to change them according to:

$$\Delta v_i^t = \beta\left[r^t\left\{y^t\frac{1-\pi^t}{\pi^t} + (1-y^t)\frac{\pi^t}{1-\pi^t}\right\} - p^t\right]x_i.$$

where π^t is an approximation to $\mathcal{E}[y^t|\mathbf{w}^t]$. p^t then estimates $\hat{b} = (1-p)r_1 + pr_0$. Since y is never 1 if $\pi^t = 0$, the first term is never infinite. However, this iterative scheme would not be expected to converge.

The alternative way, suggested by, but not discussed in, Sutton's thesis, is to develop separate predictions of r_0 and r_1, using two sets of weights. These would then be combined with π^t as $(1-\pi^t)r_1 + \pi^t r_0$. Both methods were simulated.

For the sake of comparison, I used the problems that Sutton developed for his thesis [2]. The set chosen are the non-associative ones in Chapter II, although the new comparison term will work for associative tasks too. Table 1, copied from P18, shows the problems. The binary tasks produce reinforcement of ± 1, with the probability that it is 1 given in the last two columns of the table. The continuous tasks produce reinforcement spread uniformly within $\pm$ 0.1 of the means given in the last two columns.

Formal descriptions of the algorithms compared are given in table 2, using Sutton's notation. Algorithms $\mathcal{A}$ and $\mathcal{A}'$ are Sutton's algorithms 8 and 9, which he found to be the best. $\mathcal{B}$, $\mathcal{B}'$, $\mathcal{C}$ and $\mathcal{C}'$ all make p^t estimate the quantity recommended by the analysis above. $\mathcal{B}$ and $\mathcal{B}'$ do this through a single term, whereas $\mathcal{C}$ and $\mathcal{C}'$ also employ u_1^t and u_2^t, which are designed to predict r_1 and r_0 respectively.

Table 1: The Tasks (From Sutton).

Task #	Reinforcement Type	r range	r mean Act 1	r mean Act 0
1	Binary	$\{1,-1\}$	0.90	0.80
2	Binary	$\{1,-1\}$	0.20	0.10
3	Binary	$\{1,-1\}$	0.55	0.45
4	Continuous	$\Re$	0.90	0.80
5	Continuous	$\Re$	-0.80	-0.90
6	Continuous	$\Re$	0.05	-0.05

Figures 1-6 show how the algorithms performed on each of the various tasks, for differing values of α. Figure 7 shows how the algorithms performed across the entire range of tasks, choosing for each its best result. The y-axis shows the terminal probability of choosing action 1, which is the better action for all of the tasks. It is apparent that $\mathcal{C}$ which uses the new estimator, does indeed perform better than $\mathcal{A}$ and $\mathcal{A}'$ which use the original one, although not by much. $\mathcal{B}$ and $\mathcal{B}'$ are particularly bad on the two tasks for which reinforcement is generally negative whichever action is taken. It is unclear why this only happens for these particular tasks, although dividing by π^t or $(1-\pi^t)$ does build in an instability. The obvious way to cure this - multiplying the rule by $\pi^t(1-\pi^t)$ does not improve matters substantially.

In a further experiment, the standard deviation σ of the distribution of $\eta[t]$ was set to 0.5. This value determines the balance between the exploitation of the current weight $w[t]$, and the exploration for a better one. Figure 8 is the equivalent of figure 7 for this case, showing the best performance of the algorithms, and again algorithm $\mathcal{C}$ can be seen to be somewhat superior. Indeed, it affords more improvement in this case. It is also unclear why $\mathcal{C}$ should outperform $\mathcal{C}'$, since Sutton generally found algorithms with eligibility terms of the form $y-\mathcal{E}[y]$ were preferable to those employing $y-1/2$.

A further alternative is to develop explicit estimators of $(1-p)r_1$ and pr_0, and to use their sum. In the non-associative case the resulting algorithms would not differ greatly from $\mathcal{C}$ and $\mathcal{C}'$. They would differ in the associative case, however, since the learning rule for these estimators would not change, whereas the equivalents of $\mathcal{C}$ and $\mathcal{C}'$ would involve estimators of $r_1(\mathbf{x})$ and $r_0(\mathbf{x})$, which do depend on the input $\mathbf{x}$.

4 CONCLUSIONS

At least one of the ways in which reinforcement comparison works is by reducing the effects of the non-linearity of the function which relates the weights of a stochastic learning automaton to the expected reinforcement. This is not apparent from the first or-

Table 2: The Algorithms (After Sutton).

Algorithm	Update Rule
$\mathcal{A}$	$\Delta w[t] = \alpha(r[t+1]-p[t])(y[t]-\frac{1}{2})$
$\mathcal{A}'$	$\Delta w[t] = \alpha(r[t+1]-p[t])(y[t]-\pi[t])$
$\mathcal{B}$	$\Delta w[t] = \alpha(r[t+1]-q[t])(y[t]-\frac{1}{2})$
$\mathcal{B}'$	$\Delta w[t] = \alpha(r[t+1]-q[t])(y[t]-\pi[t])$
$\mathcal{C}$	$\Delta w[t] = \alpha(r[t+1]-s[t])(y[t]-\frac{1}{2})$
$\mathcal{C}'$	$\Delta w[t] = \alpha(r[t+1]-s[t])(y[t]-\pi[t])$

Where: $\Delta w[t] \equiv w[t+1]-w[t]$, and

$w[0]=0, \pi[0]=\frac{1}{2}, y[t]\in\{1,0\}, \alpha>0,$

and $\pi[t]$ is the probability that $y[t]=1$.

For all algorithms, $y[t] = \begin{cases} 1, & \text{if } w[t]+\eta[t]>0; \\ 0, & \text{otherwise,} \end{cases}$

where $\eta[t]$ is normally distributed $\mathcal{N}[\mu=0,\sigma=0.3]$

For $\mathcal{A}$ and $\mathcal{A}'$,

$\Delta p[t] = \beta(r[t+1]-p[t]), \qquad p[0]=r[1],$

For $\mathcal{B}$ and $\mathcal{B}'$,

$\Delta q[t] = \beta\{ r[t+1]\left(\frac{(1-y[t])\pi[t]}{1-\pi[t]}+\frac{y(t)(1-\pi[t])}{\pi[t]}\right) - q[t]\}, \qquad q[0]=r[1],$

For $\mathcal{C}$ and $\mathcal{C}'$,

$\Delta s[t] = \beta(u_1[t+1](1-\pi[t])+u_0[t+1]\pi[t] - s[t]), \qquad s[0]=r[1],$

$\Delta u_1[t] = \beta(r[t+1]-u_1[t])y[t], \qquad u_1[0]=r[1],$

$\Delta u_0[t] = \beta(r[t+1]-u_0[t])(1-y[t]), \quad u_0[0]=r[1],$

and $\beta=0.2$.

All algorithms are run for 25 iterations (Sutton used 200), and each mark on the graphs in figures 1-7 is the average over 500 runs.

der term, from which one can only conclude that the reinforcement comparison algorithms are performing stochastic gradient ascent, independent of the actual comparison adopted. The second order term also reveals an optimum value for this comparison, and simulations have confirmed that the new term speeds learning, although it does not make for a dramatic improvement.

This analysis, like Williams', says nothing about the convergence of the algorithms. However, Sutton's simulations do provide some grounds for optimism.

Acknowledgements

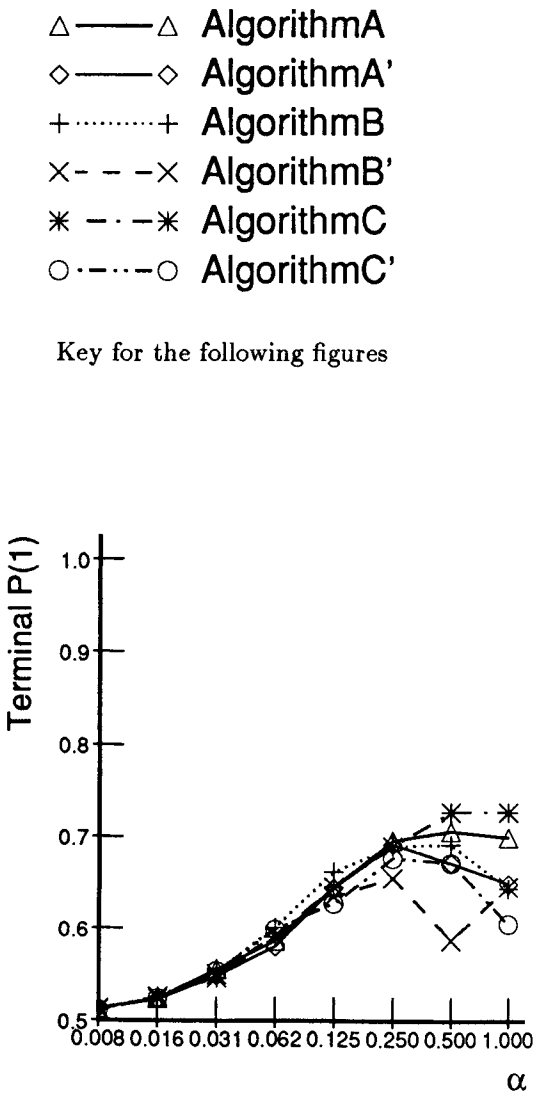
I am very grateful to Andy Barto, Geoff Hinton, Rich Sutton, Ron Williams, and David Willshaw for their helpful comments, to the students and faculty of the Summer School for the ambience, and to the SERC for their money. Part of this work was done at the University of Massachusetts at Amherst, and I particularly thank Andy Barto for his hospitality.

References

[1] Barto, AG (1985). Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, 4:229-256.

[2] Sutton, RS (1984). *Temporal Credit Assignment in Reinforcement Learning.* PhD Thesis. University of Massachusetts, Amherst, MA.

[3] Williams, RJ (1988). *Toward a theory of reinforcement - learning connectionist systems.* Technical Report NU-CCS-88-3, College of Computer Science, Northeastern University, 360 Huntingdon Avenue, Boston, MA.



Key for the following figures

Figure 1: Task 1

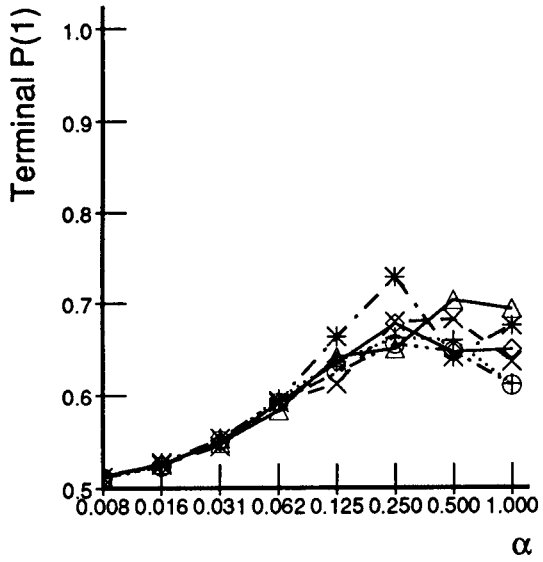


Figure 2: Task 2

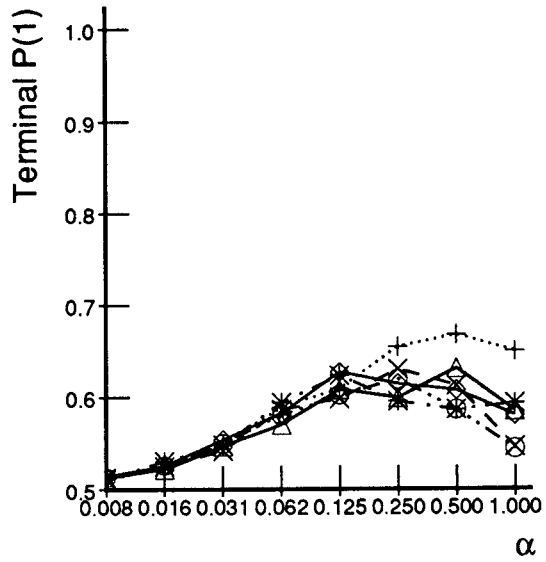


Figure 3: Task 3

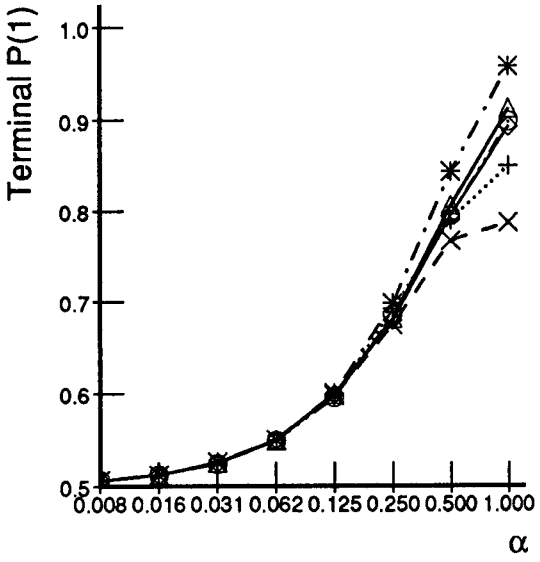


Figure 4: Task 4

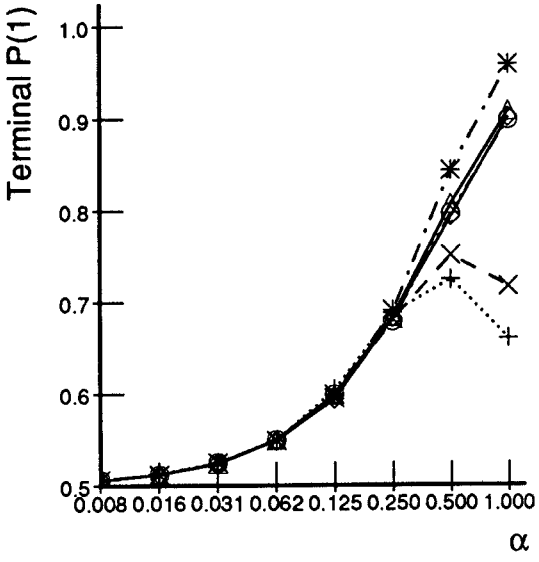


Figure 5: Task 5

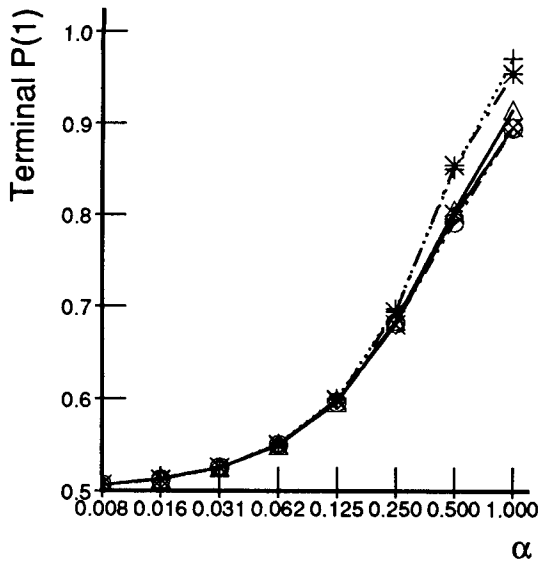


Figure 6: Task 6

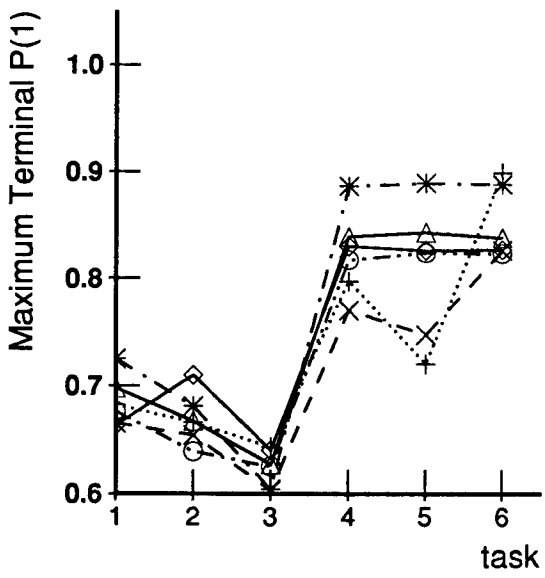


Figure 8: All tasks - best performing α, $\eta[t] \sim \mathcal{N}[0, 0.5]$

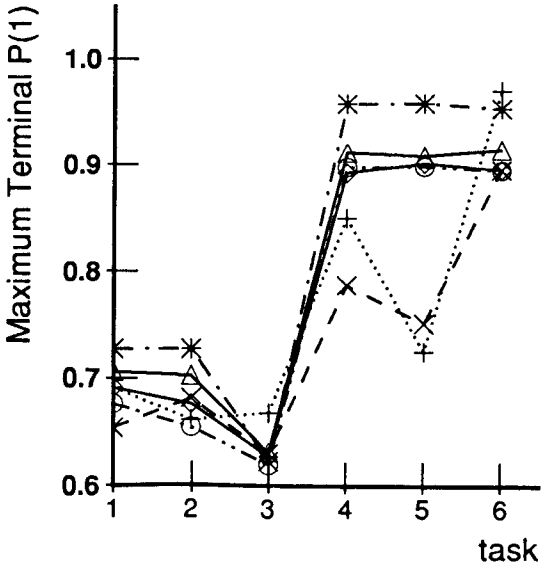


Figure 7: All tasks - best performing α