Improving Generalisation for Temporal Difference Learning: The Successor Representation

Peter Dayan
Computational Neurobiology Laboratory
The Salk Institute
PO Box 85800, San Diego CA 92186-5800

Abstract

Estimation of returns over time, the focus of temporal difference (TD) algorithms, imposes particular constraints on good function approximators or representations. Appropriate generalisation between states is determined by how similar their successors are, and representations should follow suit. This paper shows how TD machinery can be used to learn such representations, and illustrates, using a navigation task, the appropriately distributed nature of the result.

1 Introduction

The method of temporal differences (TD, Samuel 1959; Sutton, 1984; 1988) is a way of estimating future outcomes in problems whose temporal structure is paramount. A paradigmatic example is predicting the long term discounted value of executing a particular policy in a finite Markovian decision task. The information gathered by TD can be used to improve policies in a form of asynchronous dynamic programming (DP) (Watkins, 1989; Barto, Sutton & Watkins, 1990; Barto, Bradtke & Singh, 1991).

As briefly reviewed in the next section, TD methods apply to a learning *framework* which specifies the goal for learning and precisely how the system fails to attain this goal in particular circumstances. Just like the proposal to minimise mean square error, TD methods lie at the heart of different *mechanisms* operating over diverse *representations.* Representation is key – difficult problems can be rendered trivial if looked at in the correct way. It is particularly important for systems to be able to learn appropriate representations, since it is rarely obvious from the outset exactly what they should be. For static tasks, generalisation is typically sought by awarding similar representations to states that are nearby in some space. This concept extends to tasks involving prediction over time, except that adjacency is defined in terms of similarity of the future course of the behaviour of a dynamical system.

Section 3 suggests a way, based on this notion of adjacency, of learning representations that should be particularly appropriate for problems to which TD techniques have been applied. Learning these representations can be viewed as a task itself amenable to TD methods, and so requires no extra machinery. Section 4 shows the nature of the resulting representation for a simple navigation task. Part of this work was reported in Dayan (1991a; 1991b).

2 TD Learning

Consider the problem of estimating expected terminal rewards, or returns, in a finite absorbing Markov chain; this was studied in the context of TD methods by Sutton (1988). An agent makes a transition between non-absorbing states i and $j \in \mathcal{N}$ according to the ij^{th} element of the Markov matrix Q, or to absorbing state $k \in \mathcal{T}$ with probability s_{ik}, with a stochastic reinforcement or return whose mean is $\bar{z}_k$ and whose variance is finite. In this and the next section, the returns and transition probabilities are assumed to be fixed. The *immediate* expected return from state $i \in \mathcal{N}$, represented as the i^{th} element of a vector $\mathbf{h}$, is the sum of the probabilities of making immediate transitions to absorbing states times the expected returns from those states:

$$[\mathbf{h}]_i = \sum_{k \in \mathcal{T}} s_{ik} \bar{z}_k$$

The *overall* expected returns, taking account of the possibility of making transitions to non-absorbing states first, are:

$$\begin{aligned} [\bar{\mathbf{r}}]_i &= [\mathbf{h}]_i + [Q\mathbf{h}]_i + \left[Q^2\mathbf{h}\right]_i + \dots \\ &= \left[(I - Q)^{-1}\mathbf{h}\right]_i \end{aligned} \tag{1}$$

where I is the identity matrix.

The agent estimates the overall expected return from each state (compiled into a vector $\bar{\mathbf{r}}$) with a vector-valued function $\hat{\mathbf{r}}(\boldsymbol{w})$ which depends on a set of parameters $\boldsymbol{w}$ whose values are determined during the course of learning. If the agent makes the transition from state i_t to i_{t+1} in one observed sequence, TD(0) specifies that w should be changed to reduce the error:

$$\epsilon_{t+1} = [\hat{r}(w)]_{i_{t+1}} - [\hat{r}(w)]_{i_t} \tag{2}$$

where, for convenience, $[\hat{r}(w)]_{i_{t+1}}$ is taken to be the delivered return $z_{i_{t+1}}$ if i_{t+1} is absorbing. This enforces a kind of consistency in the estimates of the overall returns from successive states, which is the whole basis of TD learning. More generally, information about the estimates from later states $[\hat{r}(w)]_{i_{t+s}}$ for $s > 1$ can also be used, and Sutton (1988) defined the TD(λ) algorithm which weighs their contributions exponentially less according to λ^s.

With the TD algorithm specifying how the estimates should be manipulated in the light of experience, the remaining task is one of function approximation. How w should change to minimise the error ϵ_{t+1} in equation 2 depends on exactly how w determines $[\hat{r}(w)]_{i_t}$. Sutton (1988) represented the non-absorbing states with real-valued vectors $\{x_i\}$, $[\hat{r}(w)]_i$ as the dot product $w.x_i$ of the state vector with w taken as a vector of weights, and changed w in proportion to

$$-(w.x_{i_{t+1}} - w.x_{i_t})x_{i_t}$$

using $z_{i_{t+1}}$ instead of $w.x_{i_{t+1}}$ if i_{t+1} is absorbing. This is that part of the gradient $-\nabla_w \epsilon_{t+1}$ that comes from the error at step x_{i_t}, ignoring the contribution from $x_{i_{t+1}}$ (Werbos, 1990; Dayan, 1992).

In the 'batch-learning' case for which the weights are updated only after absorption, Sutton showed that if the learning rate is sufficiently small and the vectors representing the states are linearly independent, then the expected values of the estimates converge appropriately. Dayan (1992) extended this proof to show the same was true of TD(λ) for $0 < \lambda < 1$.

3 Time-based Representations

One of the key problems with TD estimation, and equivalently with TD based control (Barto, Sutton and Watkins, 1989), is the speed of learning. Choosing a good method of function approximation, which amounts in the linear case to choosing good representations for the states, should make a substantial difference. For prediction problems such as the one above, the estimated expected overall return of one state is a biased sum of the estimated expected overall returns of its potential successors. This implies that for approximation schemes that are linear in the weights w, a good representation for a state would be one that resembles the representations of its successors, *ie* is only a small Euclidean distance away from them (with the degrees of resemblance being determined by the biases). In this way, the estimated value of each state can be partially based on the estimated values of those that succeed it, in a way made more formal below.

For conventional, static, problems, received wisdom holds that distributed representations perform best, so long as the nature of the distribution somehow conforms with the task – nearby points have nearby solutions. The argument above suggests that the same is true for dynamic tasks, except that neighbourliness is defined in terms of temporal succession. If the transition matrix of the chain is initially unknown, this representation will have to be learned directly through experience.

Starting at state $i \in \mathcal{N}$, imagine trying to predict the expected future occupancy of all other states. For the j^{th} state, $j \in \mathcal{N}$, this should be:

$$\begin{aligned} [\bar{\mathbf{x}}_i]_j &= [I]_{ij} + [Q]_{ij} + [Q^2]_{ij} + \ldots \\ &= [(I - Q)^{-1}]_{ij}. \end{aligned} \tag{3}$$

where $[M]_{ij}$ is the ij^{th} element of matrix M and I is the identity matrix. Representing state i using $\bar{\mathbf{x}}_i$ is called the *successor representation* (SR).

A TD algorithm itself is one way of learning SR. Consider a punctate representation which devotes one dimension to each state and has the l^{th} element of the vector representing state k, $[\mathbf{x}_k]_l$, equal to $[I]_{kl}$. Starting from $i_t = i$, the prediction of how often $[\mathbf{x}_{i_s}]_j = 1$ for $s \geq t$ is exactly the prediction of how often the agent will visit state j in the future starting from state i, and should correctly be $[\bar{\mathbf{x}}_i]_j$. To learn this, the future values of $[\mathbf{x}_{i_s}]_j$ for $s \geq t$ can be used in just the same way that the future delivery of reinforcement or return in used in standard TD learning.

For a linear function approximator, it turns out that SR makes easy the resulting problem of setting the optimal weights w^* which are defined as those making $\bar{\mathbf{r}} = \hat{\mathbf{r}}(w^*)$. If $\bar{X}$ is the matrix of vectors representing the states in the SR, $[\bar{X}]_{ij} \equiv [\bar{\mathbf{x}}_j]_i$, then w^* is determined as:

$$\bar{X}^T w^* = \bar{\mathbf{r}}, \text{ which implies, from equations 1 and 3, that}$$

$$w^* = h.$$

But h is just the expected immediate return from each state – it is insensitive to all the temporal dependencies that result from transitions to non-absorbing states.

The SR therefore effectively factors out the entire temporal component of the task, leaving a straightforward estimation problem for which TD methods would not be required. This can be seen in the way that the transition matrix Q disappears from the update equation, just as would happen for a non-temporal task without a transition matrix at all. For instance, for the case of an absorbing Markov chain with batch-learning updates, Sutton showed that the TD(0) update equation for the mean value of the weights $\bar{w}_n$ satisfies

$$\bar{w}_{n+1} = \bar{w}_n + \alpha XD(h + QX^T\bar{w}_n - X^T\bar{w}_n)$$

where X is the representation, α is the learning rate and, since the updates are made after observing a whole sequence of transitions from start to absorption rather than just a single one, D is the diagonal matrix whose diagonal elements are the average number of times each state is visited on each sequence. Alternatively, directly from the estimates of the values of the states,

$$(X^T\bar{w}_{n+1} - \bar{r}) = (I - \alpha X^T XD(I - Q))(X^T\bar{w}_n - \bar{r}),$$

Using $\bar{X}$ instead, the update becomes:

$$\begin{aligned} \bar{w}_{n+1} &= \bar{w}_n + \alpha\bar{X}D(h - \bar{w}_n), \text{ or} \\ (\bar{w}_{n+1} - h) &= (I - \alpha\bar{X}D)(\bar{w}_n - h). \end{aligned}$$

Since $\bar{X}$ is invertible, Sutton's proof that $\bar{X}^T\bar{w}_n \to \bar{r}$, and therefore that $\bar{w}_n \to h$ as $n \to \infty$, still holds. I conjecture that the variance of these estimates will be lower than those for other representations X (*eg* $X = I$) because of the exclusion of the temporal component.

For control problems it is often convenient to weigh future returns exponentially less according to how late they arrive – this effectively employs a discount factor. In this case the occupancy of future states in equation 3 should be weighed exponentially less by exactly the same amount.

A possible objection to using TD learning for SR is that it turns the original temporal learning problem – that of predicting future reinforcement – into a whole set of temporal learning problems – those of predicting the future occupancy of all the states. This objection is weakened in two cases:

- The learned predictions can be used merely to *augment* a standard representation such as the punctate one. An approximately appropriate representation can be advantageous even before all the predictions are quite accurate. Unfortunately this case is hard to analyse because of the interaction between the learning of the predictions and the learning of the returns. Such a system is used in the navigation example below.

- The agent could be allowed to learn the predictions by exploring its environment before it is first rewarded or punished. This can be viewed as a form of latent learning and works since the representation does not depend on the returns.

One could regard these predictions as analogous to the *hidden* representations in Anderson's (1986) multi-layer backpropagation TD network in that they are fashioned to be appropriate for learning TD predictions but are not directly observable and so have to be learned. Whereas Anderson's scheme uses a completely general technique which makes no explicit reference to states' successors, SR is based precisely on what should comprise a good representation for temporal tasks.

4 Navigation Illustration

Learning the shortest paths to a goal in a maze such as the one in figure 1 was chosen by Watkins (1989) and Barto, Sutton & Watkins (1989) as a good example of how TD control works. For a given policy, *ie* mapping from positions in the grid to directions of motion, a TD algorithm is used to estimate the distance of each state from the goal. The agent is provided with a return of -1 for every step that does not take it to the goal and future returns, *ie* future steps, are weighed exponentially less using a discount factor. The policy is improved in an asynchronous form of dynamic programming's policy iteration by making more likely those actions whose consequences are better than expected.

Issues of representation are made particularly clear in such a simple example. For the punctate case, there can be no generalisation between states. Distributed representations can perform better, but there are different methods with different qualities. Watkins (1989), for a similar task, used a representation inspired by Albus' CMAC (1976). In this case, CMAC squares which cover patches of 3×3 grid points are placed regularly over

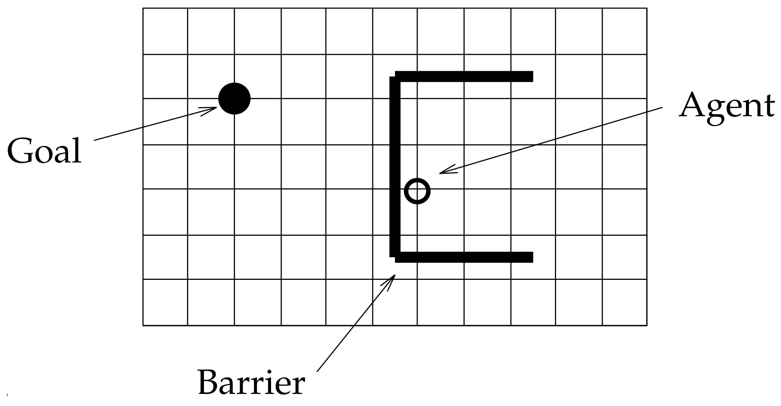

Figure 1: The grid task. The agent can move one step in any of the four directions except where limited by the barrier or by the walls.

---

the grid such that each interior grid point is included in 9 squares. The output of the units corresponding to the squares is 0 if the agent is outside their receptive fields, and otherwise is modulated by the distance of the agent from the centre of the relevant square. Over most of the maze this is an excellent representation – locations that are close in the Manhattan metric on the grid are generally similar distances from the goal, and are also covered by many of the same CMAC squares. Near the barrier, however, the distribution of the CMACs actually hinders learning – locations close in the grid but on opposite sides of the barrier are very different distances from the goal, and yet still share a similar CMAC square representation.

By contrast, the successor representation, which was developed in the previous section, produces a CMAC-like representation that adapts correctly to the barrier. If the agent explores the maze with a completely random policy before being forced to find the goal, the learned SR would closely resemble the example shown in figure 2. Just like a CMAC square, the representation decays exponentially away from the starting state $(5, 6)$ in a spatially ordered fashion – however note SR's recognition that states on the distant side of the barrier are actually very far away in terms of the task (and so the predictions are too small to be visible). Simulations confirm that using the SR in conjunction with a punctate respresentation leads to faster learning for this simple task (see figure 3), even if the agent does not have the chance to explore the maze before being forced to find the goal.

This example actually violates the stationarity assumption made in section 2, that transitions probabilities and returns are fixed. As the agent improves its policy, the mean

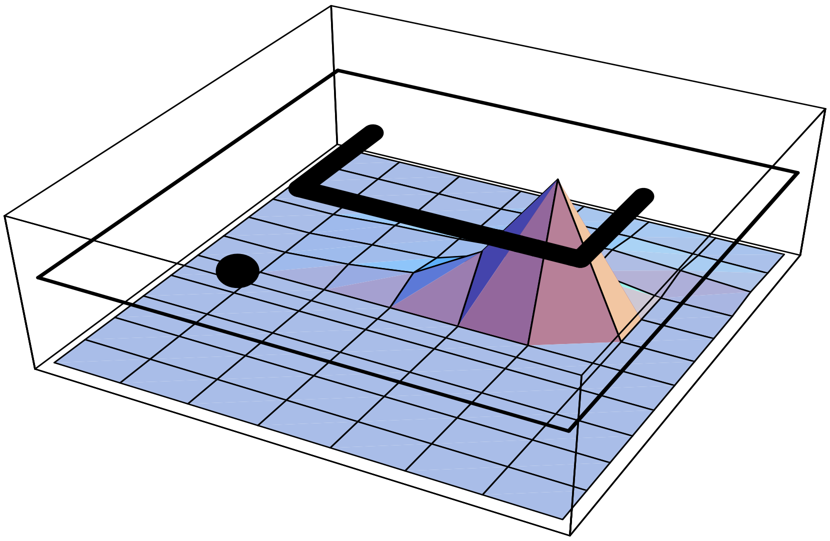

Figure 2: The predictions of future occupancy starting from (5,6) after exploration in the absence of the goal. The z-coordinate shows the (normalised) predictions, and the barrier and the goal are overlaid. The predictions decay away exponentially from the starting location, except across the barrier.

---

number of steps it takes to go from one state to another changes, and so SR should change too. Once the agent moves consistently along the optimal path to the goal, locations that are not on it are never visited, and so the prediction of future occupancy of those should be 0. Figure 4 shows the difference between the final and initial sets of predictions of future occupancy starting from the same location $(5,6)$ as before. The exponential decay along the path is caused by the discount factor. The path taken by the agent is clear. If the task for the agent were changed such that it had to move from anywhere on the grid to a different goal location, this new form of the SR would actually hinder the course of learning, since its distributed character no longer correctly reflects the actual nature of the space. This demise is a function of the linked estimation and control, and would not be true for pure estimation tasks.

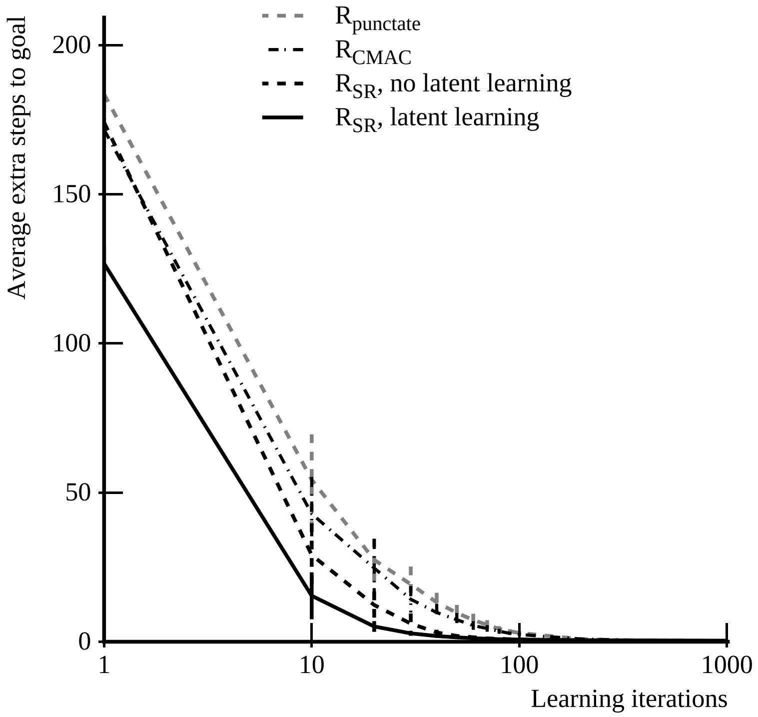

Figure 3: Learning curves comparing punctate representation ($R_{punctate}$), CMAC-squares (R_{CMAC}) and a punctate representation augmented with the SR (R_{SR}), in the latter case both with and without an initial, un-rewarded, latent learning phase. TD control learning as in Barto, Sutton and Watkins (1989) is temporarily switched off after the number of trials shown in the $x-$axis, and the $y-$axis shows the average number of excess steps the agent makes on the way to the goal starting from every location in the grid. Parameters are in Dayan (1991b).

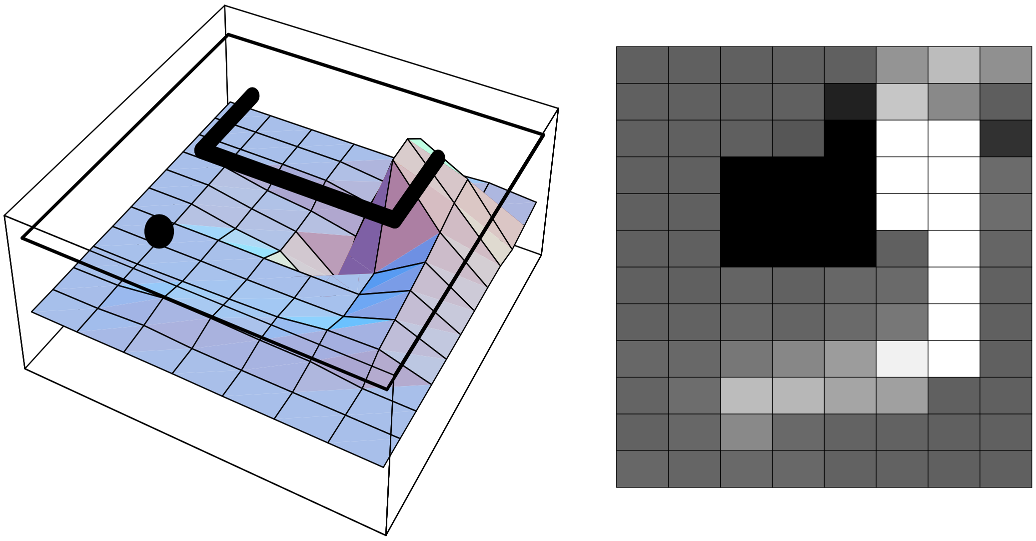

Figure 4: The degradation of the predictions. Both graphs show the differences between the predictions after 2000 steps and those initially – the left graph as a surface, with the barrier and the goal overlaid, and the right graph as a density plot. That the final predictions just give the path to the goal is particularly clear from the white (positive) area of the density plot – the black (negative) area delineates those positions on the grid that are close to the start point $(5, 6)$, and therefore featured in the initial predictions, but are not part of this ultimate path.

5 Discussion

This paper has considered some characteristics of how representation determines the performance of TD learning in simple Markovian environments. It suggests that what amounts to a local kernel for the Markov chain is an appropriate distributed representation, because it captures all the necessary temporal dependencies. This representation can be constructed during a period of latent learning and is shown to be superior in a simple navigation task, even over others that also share information between similar locations.

Designing appropriate representations is a key issue for many of the sophisticated learning control systems that have recently been proposed. However, as Barto, Bradtke and Singh (1991) pointed out, a major concern is that the proofs of convergence of TD learning have not been very extensively generalised to different approximation methods.

Both Moore (1990) and Chapman and Kaelbling (1991) sought to exorcise the dæmon of dimensionality by using better function approximation schemes, which is an equivalent step to using a simple linear scheme with more sophisticated input representations. Moore used kd trees (see Omohundro, 1987, for an excellent review), which have the added advantage of preserving the integrity of the actual values they are required to store, and so preserve the proofs of the convergence of $\mathcal{Q}-$learning (Barto, Bradtke & Singh, 1991; Watkins & Dayan, 1992). However just like the CMAC representation described above, the quality of the resulting representation depends on an *a priori* metric, and so is not malleable to the task.

Chapman and Kaelbling also used a tree-like representation for $\mathcal{Q}-$learning, but their trees were based on logical formulæ satisfied by their binary-valued input variables. If these variables do not have the appropriate characteristics, the resulting representation can turn out to be unhelpful. It would not afford great advantage in the present case.

Sutton (1990), Thrun, Möller and Linden (1991), and others have suggested the utility of learning the complete transition matrix of the Markov chain, or, for the case of control, the mapping from states and actions to next states. Sutton used this information to allow the agent to learn while it is disconnected from the world. Thrun, Möller and Linden used it implicitly to calculate the cost of and then improve a projected sequence of actions. The SR is less powerful in the sense that it only provides an appropriately distributed representation and not a veridical map of the world. A real map has the added advantage that its information is independent of the goals and policies of the agent; however it is more difficult to learn. Sutton's scheme could equally well be used to improve a system based on the learned representation.

Sutton and Pinette (1985) discussed a method for control in Markovian domains that is closely related to the SR and which uses the complete transition matrix implicitly defined by a policy. In the notation of this paper, they considered a recurrent network effectively implementing the iterative scheme

$$\hat{x}_{n+1} = x_i + Q\hat{x}_n$$

where x_i is the punctate representation of the current state i and Q is the transition matrix. $\hat{x}_n$ converges to $\bar{x}_i$ from equation 3, the SR of state i. Rather than use this for representational purposes, however, Sutton and Pinette augmented Q so that the sum of future returns is directly predicted through this iterative process. This can be seen as an alternative method of eliminating the temporal component of the task, although the use of the recurrence implies that the the final predictions are very sensitive to errors in the estimate of Q.

The augmented Q matrix is learned using the discrepancies between the predictions at adjacent time steps – however the iterative scheme complicates the analysis of the convergence of this learning algorithm. A particular advantage of their method is that a small change in the model (*eg* a slight extension to the barrier) can instantaneously lead to dramatic changes in the predictions. Correcting the SR would require relearning *all* the affected predictions explicitly.

Issues of representation and function approximation are just as key for sophisticated as unsophisticated navigation schemes. Having a representation that can learn to conform to the structure of a task has been shown to offer advantages – but any loss of the guarantee of convergence of the approximation and dynamic programming methods is, of course, a significant concern.

Acknowledgements

I am very grateful to Read Montague, Steve Nowlan, Rich Sutton, Terry Sejnowski, Chris Watkins, David Willshaw, the connectionist groups at Edinburgh and Amherst, and the large number of people who read drafts of my thesis, for their help and comments. I am especially grateful to Andy Barto for his extensive and detailed criticism and for pointers to relevant literature. Support was from the SERC.

References

[1] Albus, JS (1975). A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC). *Transactions of the ASME: Journal of Dynamical Systems, Measurement and Control,* **97**, pp 220-227.

[2] Anderson, CW (1986). *Learning and Problem Solving with Multilayer Connectionist Systems.* PhD Thesis, University of Massachusetts, Amherst, MA.

[3] Barto, AG, Bradtke, SJ & Singh, SP (1991). *Real-Time Learning and Control using Asynchronous Dynamic Programming.* TR 91-57, Department of Computer Science, University of Amherst, MA.

[4] Barto, AG, Sutton, RS & Watkins, CJCH (1989). *Learning and Sequential Decision Making.* Technical Report 89-95, Computer and Information Science, University of Massachusetts, Amherst, MA.

[5] Chapman, D & Kaelbling, LP (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence.* To appear.

[6] Dayan, P (1991a). Navigating through temporal difference. In RP Lippmann, JE Moody & DS Touretzky, editors, *Advances in Neural Information Processing, 3,* pp 464-470. San Mateo, CA: Morgan Kaufmann.

[7] Dayan, P (1991b). *Reinforcing Connectionism: Learning the Statistical Way.* PhD Thesis, University of Edinburgh, Scotland.

[8] Dayan, P (1992). The convergence of TD(λ) for general λ. *Machine Learning,* **8**, 341-362.

[9] Moore, AW (1990). *Efficient Memory-based Learning for Robot Control.* PhD Thesis, University of Cambridge Computer Laboratory, Cambridge, England.

[10] Omohundro, S (1987). Efficient algorithms with neural network behaviour. *Complex Systems,* **1**, pp 273-347.

[11] Samuel, AL (1959). Some studies in machine learning using the game of checkers. Reprinted in EA Feigenbaum and J Feldman, editors, *Computers and Thought.* McGraw-Hill (1963).

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

[13] Sutton, RS (1988). Learning to predict by the methods of temporal difference. *Machine Learning,* **3**, pp 9-44.

[14] Sutton, RS (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning.* San Mateo, CA: Morgan Kaufmann.

[15] Sutton, RS & Pinette, B (1985). The learning of world models by connectionist networks. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society,* pp 54-64. Irvine, CA: Lawrence Erlbaum.

[16] Thrun, SB, Möller, K & Linden, A (1991). Active exploration in dynamic environments. In RP Lippmann, JE Moody & DS Touretzky, editors, *Advances in Neural Information Processing, 3,* pp 450-456. San Mateo, CA: Morgan Kaufmann.

[17] Watkins, CJCH (1989). *Learning from Delayed Rewards.* PhD Thesis. University of Cambridge, England.

[18] Watkins, CJCH & Dayan, P (1992). Q-learning. *Machine Learning,* **8**, pp 279-292.

[19] Werbos, PJ (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks,* **3**, pp 179-189.