

Trading off Perception with Internal State: Reinforcement Learning and Analysis of Q-Elman Networks in a Markovian Task

Bram Bakker

Gwendid van der Voort van der Kleij

Unit of Experimental and Theoretical Psychology; Leiden University
P.O. Box 9555; 2300 RB, Leiden, The Netherlands
{bbakker, st952508}@fsw.leidenuniv.nl

Abstract

A Markovian reinforcement learning task can be dealt with by learning a direct mapping from states to actions or values, or from state-action pairs to values. However, this may involve a difficult pattern recognition problem when the state space is large. This paper shows that using internal state, called “supportive state”, may alleviate this problem—presenting an argument against the tendency to almost automatically use a direct mapping when the task is Markovian. This point is demonstrated in simulation experiments of an agent controlled by a neural network capable of learning the strategy of direct mapping as well as internal state, combining $Q(\lambda)$ -learning and recurrent neural networks in a new way. The trade-off between the two strategies is investigated in more detail, focusing in particular on border cases.

1 Introduction

Reinforcement learning is a broad class of methods that allows agents to learn intelligent behavior in complex environments on the basis of (delayed) scalar reward signals. An environment can be anything from a physical environment such as a maze to an abstract scheduling task. Most reinforcement learning work is concerned with Markovian tasks, or Markov Decision Problems. In a Markovian task the input to the agent is an accurate reflection of the complete state of the environment. After an agent’s action, the response (or probability distribution of responses) from the environment, consisting of a reward and the next state, depends only on the current state perceived by the agent and the action, and not on the history of states and actions or any other information.

Because of this, almost all work on reinforcement learning in Markovian tasks focuses on learning a *direct mapping* from perceived states to actions [1] or values [13], or from state-action pairs to values (Q-learning [16] and SARSA [12, 14]). The direct mapping or purely perception-based strategy can be implemented using a table [16, 15] or using a function approximator, such as a feedforward neural network [1, 12, 7]. Only when the task is non-Markovian is it necessary to equip the agent with some sort of *internal state*. This internal state represents the history of inputs and actions [10, 9, 8] or it contains a more explicit model [3] of the environment. It may be implemented using a recurrent neural network [9, 8].

However, even in a Markovian task, the single job of learning the direct mapping can be difficult. One often investigated problem results from dealing with a large state space. Obtaining sufficient experience with every single state may take too much time, and generalization over the state space is necessary. That is the problem that this paper is concerned with. Usually, in these cases the agent’s input is a vectored representation of the environment’s state. A direct mapping from states or state-action pairs to actions or values which generalizes in the right way then requires complex pattern recognition. Since feedforward neural networks can learn such complex direct mappings, many researchers have used them in this context [7, 1, 12].

The main idea behind this paper is that even in Markovian tasks, the pattern recognition problem may become so difficult that it makes sense to (perhaps partially) sidestep this problem by learning to exploit information from previous time steps. In other words, in certain environments it may be easier for the agent to develop an internal state that helps the agent deal with the environment, than it is for the agent to use only perception of the current state and solve the full pattern recognition problem. We call this type of internal state *supportive state*, because even though this internal state is not strictly *necessary* to solve the (Markovian) task, it provides extra information that *supports* the agent in choosing its actions. Note that this internal state is not simply given to the agent. The idea is that even when the agent first has to learn this internal state before it can exploit it, this can still be more efficient than learning just the direct mapping.

This paper presents a concrete demonstration or “proof of principle” of this idea and in that way makes a case against the strong tendency in the reinforcement learning community to almost automatically use direct mappings when the task is Markovian [6, 15]. Similarly, researchers in behavior-based robotics, autonomous agents, and the dynamic systems approach have advocated purely “reactive” systems, “direct perception-action coupling”, etc., and often argue against “representations” [2, 4]. To a lesser extent, this paper can also be read to argue, in general, that even when a task can in principle be solved by a purely reactive system, internal state may be useful.

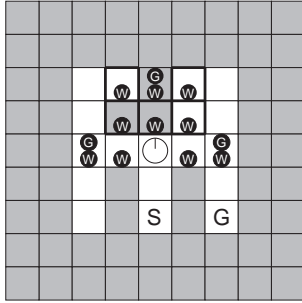


Figure 1: Example of a 5-parity maze. The agent is at the central T-junction, oriented to the north. Black circles denote its sensors for walls (W) and for the goal (G). Bold lines indicate the area encoding the parity pattern 10011, which can be perceived using wall sensors. S is the starting position and G is the goal.

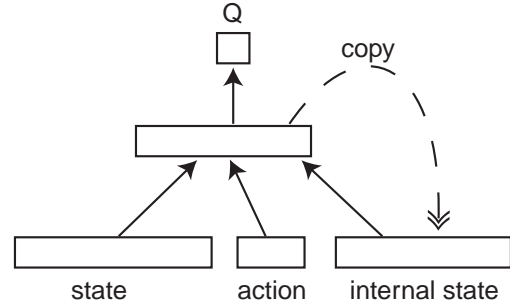


Figure 2: Q-Elman neural network. The solid lines indicate fully connected, learning weights. The dashed line represents the copy operation of the hidden units to the context units.

A concrete demonstration requires an adaptive system which can adopt either of the two strategies, the purely perception-based strategy versus the internal state strategy, depending on the specific task at hand. For this reason, simulation experiments were performed of an agent controlled by a neural network capable of learning either strategy, combining Q-learning and recurrent neural networks in a new way. Furthermore, the trade-off between the two strategies is investigated in more detail, focusing in particular on border cases.

The next section describes the setup of the simulation experiments, and it points out similarities with other work. Section 3 presents the results of the simulations and an analysis of the results. Section 4, finally, contains a general discussion.

2 Setup of the simulation experiments

2.1 The reinforcement learning task

The simulated agent’s environment is a small grid maze, consisting of open space and impenetrable walls (figure 1). The agent’s job is to move to the goal position, starting from the fixed starting position. The goal can be located in two possible positions, which can be referred to as “left” or “right”. In figure 1 the goal is in the right position, the left goal position is on the opposite side of the starting position. If the agent reaches the goal position, it receives a reward of 1, at all others points the reward is 0.

This reinforcement learning task was set up to combine a number of properties. For our purposes, it is necessary to have a large input space in which the agent must learn to recognize the relevant features, combined with relatively short optimal paths to the goal, such that learning does not take excessively long. For this reason, the agent has an array of 13 sensors detecting in its immediate surroundings walls versus open space and the presence of the goal (see figure 1 for the exact layout of the sensors), combined with steps of 2 grid positions at a time. The agent can be oriented to the north, east, south, or west. It has a choice of 3 actions: go forward, go left, or go right. If the agent attempts to move through a wall, it stays in the same grid position, but it may turn. Note that the agent can only indirectly derive its position in the maze from its sensory inputs.

Most importantly for our purposes, the task is Markovian, but the difficulty of the pattern recognition problem incorporated in the task can be increased or decreased by changing the pattern of walls and open space of the maze. These patterns are part of the regular input. Thus, the (varying) pattern recognition problem, which is the focus of this research, is a more or less natural ingredient of the learning task, and it is similar to that in other reinforcement learning work. We made sure that changes in the pattern of walls and open space did not affect the possible paths in the maze: these changes only affect the pattern recognition problem.

Coming from the fixed starting position, at the central T-junction in the maze the agent must make a crucial decision whether to go left to reach the left goal position or right to reach the right goal position (see figure 1). This is the point where the pattern recognition problem is actually varied. The perceptual information at that junction consists, in part, of inputs, encoded in walls and open space, that essentially present the agent with the *parity problem*. If those inputs have an even number of 1s (grid positions with open space), the goal is in the left position and the agent should go left; if the inputs have an odd number of 1s, the goal is in the right position and the agent should go right.

The parity problem provides us with a straightforward means of gradually (though not necessarily uniformly) increasing the difficulty of the pattern recognition problem by increasing N , the length of the pattern of bits. We use 5 different maze environments, with N ranging from 1 to 5. In all 5 mazes, at the start of each episode, the goal is

randomly placed either left or right. A random even parity pattern of length N is placed at the T-junction if the goal is left, and a random odd parity pattern is placed if the goal is right. In the 5-parity maze, all 5 positions indicated by bold lines in figure 1 code for the parity. In the 1-parity maze, only the upperleft grid position with bold lines is used, and the other four are freed as walls.

The parity problem is a classic and challenging problem. Using supervised learning, feedforward neural networks without a hidden layer cannot solve this task for $N > 1$, but feedforward networks with a hidden layer can [11]. $N = 1$ is still fairly easy: if the single input is 0, go left, if it is 1, go right. $N = 2$ already amounts to the XOR problem.

In all 5 mazes, at the starting position the agent can perceive (but not move to) the goal (see figure 1). On this basis, the agent may develop internal state, but it does not have to. If it does, at the central T-junction it needs not solve the pattern recognition problem, but it can use memory of the goal position instead. Thus, while the pattern recognition problem is varied, the opportunity for developing internal state is held constant over all 5 mazes.

2.2 The learning algorithm

The agent is controlled by a Simple Recurrent Network or Elman network [5]. The network essentially implements the $Q(\lambda)$ algorithm [16], and hence is called a Q-Elman network (see figure 2). The output of the network codes for the Q-value, which in this case is not only a function of state (perceptual input) and action, but also of the internal state of the network, encoded by the context units of the Q-Elman network.¹ On the basis of the Q-values for all 3 actions, the action is selected using the Boltzmann distribution [15, 7, 9, 8]. No annealing mechanism was used, the temperature τ of the Boltzmann function was fixed at .04.

A basic idea behind Q-learning is to incrementally estimate the value of each state-action pair, the Q-value, through the method of *temporal differences* [13]: rather than updating the value of a state-action pair based on all following delayed rewards, it is updated using the immediate reward and the maximally attainable Q-value at the next timestep. The optimal Q-function must satisfy

$$Q(s_t, a_t, I_t) = r + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, I_{t+1}) \quad (1)$$

where r is the immediate reward fed back from the environment after executing action a_t in state s_t and internal state I_t at time t , and γ is a discount factor which adjusts the importance of long-term consequences of actions.

$Q(\lambda)$ differs from the most basic version of the algorithm, $Q(0)$, in that the first includes *eligibility traces*, exponentially decaying traces indicating which state-action pairs have been visited in the recent past. In function approximators such as neural networks, a trace must be maintained for each parameter, in this case each weight, rather than for each state-action pair as in the tabular version [15].

To implement this in the Elman network, we use a variation of the standard backpropagation procedure [11]. Each weight $w_{i \rightarrow j}$ is updated according to

$$w_{i \rightarrow j}(t+1) = w_{i \rightarrow j}(t) + \eta \frac{\partial P}{\partial o_j} e_{i \rightarrow j}(t) \quad (2)$$

where η is the learning rate and o_j is the output of neuron j . P is the negative of the sum of squared errors. It can be computed relatively straightforwardly in this case, because the righthand side of equation (1) can, in neural network terms, be viewed as the target output. To implement eligibility traces, $\frac{\partial o_j}{\partial w_{i \rightarrow j}}$ of standard backprop is replaced by $e_{i \rightarrow j}(t)$, where

$$e_{i \rightarrow j}(t) = \gamma \lambda e_{i \rightarrow j}(t-1) + \frac{\partial o_j}{\partial w_{i \rightarrow j}} \quad (3)$$

where λ is the trace-decay parameter. If $\lambda = 0$, equation (2) collapses to the standard backprop formula, and we are left with $Q(0)$. Note that internal state is treated in the same way as state and action, as far as eligibility is concerned. Because Q-learning is a so-called off-policy method, the eligibility traces are set to 0 if an exploratory action is chosen [15].

The network is not explicitly instructed on what actions to take or what information to attend to. The network may decide to mainly use the immediate perception of the current state, basically behave as a feedforward network, and thus implement a direct mapping. Alternatively, it may decide to pay more attention to the context units and develop differentiated internal states. In either case, this choice is based on experienced success in the particular environment, one of the five N -parity mazes. Our hypothesis predicts a trade-off between perception and internal state: in N -parity

¹In any learning neural network, the changing weights can be viewed as a slowly changing internal state, since the network's response to an input changes with changing weights. Obviously, in this paper internal state refers to the kind of short-term memory provided by the context units, and not to the long-term memory provided by the learning weights.

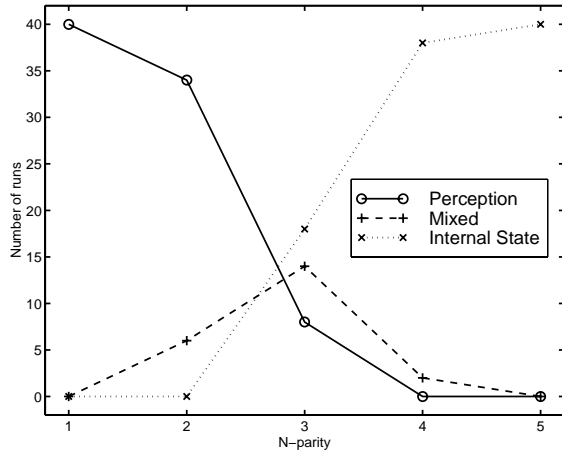


Figure 3: Strategies converged upon by the agents for each N -parity.

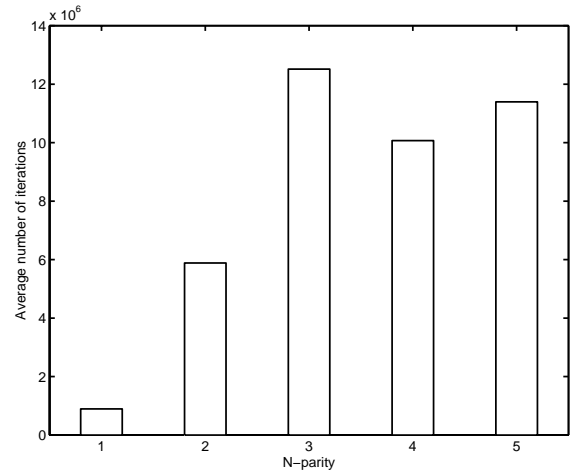


Figure 4: Average number of iterations needed to reach the termination criterion, per N -parity.

mazes with low N , agents will tend to use immediate perception, whereas in mazes with high N , agents will tend to use internal state. This would provide evidence for different strategies being the “better” ones in different Markovian tasks, in terms of ease of learning and effective use of the available information.

2.3 Related work

Feedforward neural networks are combined with $Q(\lambda)$ in [12, 7]. Watkins [16] and Sutton [14] use CMAC, another type of function approximator, in combination with $Q(\lambda)$ and SARSA(λ) respectively. At first sight, Lin & Mitchell’s [9, 8] work on reinforcement learning in non-Markovian domains seems very similar to our work, since they also combine a version of $Q(\lambda)$ with Elman networks. Once we look closer, however, there are a number of significant differences. First of all, they use a separate network for each action (as in [12, 7]), whereas we use just one network. They use backpropagation through time, whereas we use ordinary backpropagation. Finally, their specific learning algorithm (which is essentially the same as the one in [7]) allows updating only after the completion of an entire episode, and they replay memorized episodes many times, making it a memory-based approach. In contrast, our algorithm is completely online and local in time.

3 Results

3.1 Analysis of the successful strategies

For each N , 40 runs were performed. The termination criteria were as followed. Using the stochastic action selection, the agent must successfully reach the goals in more than 98% of the episodes and the running average of the number of steps needed to reach the goal must be below 3.7 (the optimal value is 3). In addition, the agent must reach the goal in 100% of the episodes when using greedy action selection ($\tau = 0$). The number of hidden nodes and context nodes was 10, $\eta = .1$, $\lambda = .9$, and $\gamma = .9$. All runs reached the termination criteria. After completion of the runs, they were individually analyzed.

To determine the strategies followed by the agents, each agent was tested in two ways. The relationship of the parity information with the goal position was reversed, so even parity now corresponds to the goal position being *right* and odd parity corresponds to *left*. If the agent now turns left where it should turn right and vice versa, this means it relies on perception of the parity information. If its performance is not affected, apparently it uses internal state. As the other, complementary test, the agent was given misleading information at the starting position, with the normal parity to goal position relationship in place. With the goal on the left side, the right goal sensor was turned on, and with the goal on the right side, the left goal sensor was turned on. Now, if the agent is misled and goes the wrong way, this means it uses internal state. If it ignores the misleading information and maintains good performance, this means it uses perception of the parity information. The results of this analysis are shown in figure 3.

All 40 agents in the 1-parity maze completely depend on perception of the parity pattern. In contrast, all 40 agents in the 5-parity maze use internal state. First of all, it is clear that agents do not develop internal state simply because the Q-Elman network has recurrent connections. Apparently, there must be a reason to develop internal state, and this reason is absent in the 1-parity maze; but obviously it is present in the 5-parity maze. In the intermediate N -parity mazes, a gradual transition from perception-based strategies to internal state-based strategies can be observed. All this confirms our hypothesis, for this setting at least, that there is a trade-off between perception and internal state.

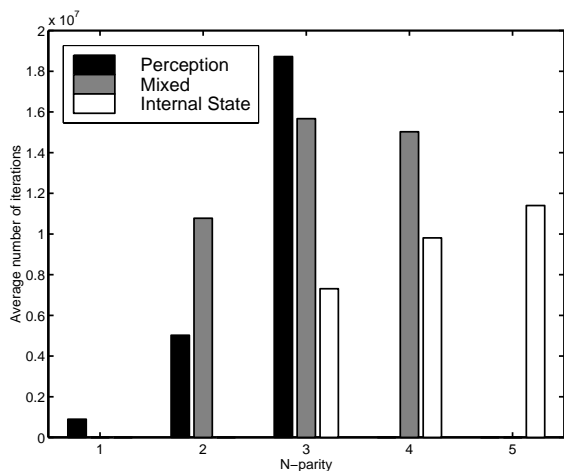


Figure 5: Average number of iterations needed to reach the termination criterion, per N -parity and per strategy.

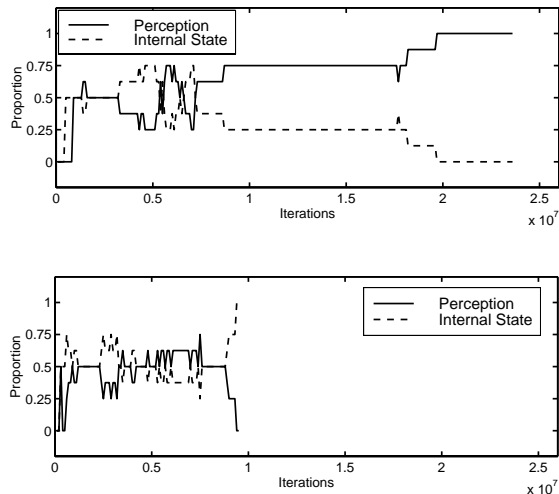


Figure 6: Development of strategy over time, in the 3-parity maze. The upper graph shows a typical run converging to perception, the lower graph a typical run converging to internal state.

If the pattern recognition problem is easy, agents tend to use perception (direct mapping), if it is hard, agents tend to use internal state. The internal state can be called supportive state in this case, because an agent could have relied on perception alone, but instead develops internal state to support its action decisions.

Interestingly, a number of agents adopt *mixed* strategies, relying on a combination of perception and internal state, with neither being sufficient to behave perfectly in either of the two tests. To explain this, we should note that as far as the hidden layer of the Elman network is concerned, both perceptual input and context unit activations appear as an input pattern. There is no compelling reason why the agent should not exploit distinctive patterns in both types of input.

In many cases, internal state-based agents developed a surprising substrategy at the starting position. When the goal is on one side, they proceed to the T-junction directly. When the goal is on the other side, however, they turn toward the goal before moving to the T-junction. This action prevents the agent from reaching the optimal policy of 3 steps. Q-learning is only guaranteed to reach the optimal policy under a number of conditions, one of which is that the Q-values must be stored in a table [16]. If function approximation is used, the usual problem of local optima applies. However, in this case, this extra action may actually be useful. We believe that it helps in creating two very different hidden unit activation patterns for each of the two goal positions, because it makes the sensory inputs and the actions very different. This results in two very different internal states that subsequently can be easily distinguished at the T-junction. Thus, this seems to be an instance of spontaneous *epistemic action*, action that is not primarily intended to accomplish a goal, but rather to facilitate the computational task faced by an agent. Humans and animals often use epistemic action to cope with difficult tasks [4].

3.2 Time needed to reach the termination criterion

Figure 4 shows the average number of iterations needed for each run, for each N -parity maze. As expected, the 1-parity maze required many less iterations than the other 4 mazes. An analysis of variance of these data found significant differences ($F = 18.312$, $p < .001$), and a post-hoc Tukey test revealed that, statistically, this result should be attributed mainly to the large difference of 1-parity and 2-parity with the others.

Nevertheless, it is somewhat surprising that the average number of iterations needed for 3-parity is larger than that for 4-parity or 5-parity. Objectively speaking, 4-parity and 5-parity are more difficult pattern recognition problems than 3-parity. Figure 5 shows the average number of iterations needed for each run in an N -parity maze, sorted by the strategy converged upon by the agent. It is apparent that the long duration of the 3-parity runs can be attributed completely to agents that learn a perception-based or mixed strategy. 3-parity seems to be a border case where both perception and internal state are viable strategies (see also figure 3). It seems plausible that at some point, an agent “opts” for one of the strategies and usually sticks to that strategy. In neural network terms, once a gradient descent network is in a certain region of weight space, it will more or less stay there. If the chosen strategy happens to be perception-based, it requires solving the 3-parity pattern recognition problem. Solving 3-parity simply takes a lot of time: focusing only on the perception-based strategies in figure 5, we see a gradual increase of learning time with

increasing N . But once the network is at that point, it cannot go back.

Evidence for this idea is presented in figure 6. Two fairly typical runs in the 3-parity maze are depicted, showing the development of the tendency toward either strategy over time. This tendency is measured as the proportion of parity patterns dealt with correctly when submitted to the two tests described in section 3.1. The lower graph shows a run converging to an internal state-based strategy, the upper graph shows a run converging to a perception-based strategy. It can be seen how the agent in the lower graph, once it has opted for internal state, quickly converges to a successful solution. The agent in the upper graph, however, still needs many iterations after having committed itself to the perception-based strategy.

In the 4-parity and 5-parity mazes, it is probably more “obvious” to the agent from the outset that it should use internal state. The correlation of the parity pattern with the goal position is even less evident than in the 3-parity case, so the parity information looks more like noise which should be ignored. Thus, the agent will not likely get stuck in a strategy trying to solve the parity problem.

4 Discussion

In the simulation experiments of agents in Markovian tasks presented here, there is a trade-off between perception and internal state. If the pattern recognition problem of perception becomes too hard, agents develop internal states, called supportive states, that reduce the reliance on perception. Of course, the environments used in this paper were constructed in a way that made developing such internal states possible and even relatively easy. Nevertheless, this work presents a proof of principle that argues against the tendency in the reinforcement learning community to consider the use of direct mappings from states or state-action pairs to actions or values as a given when the task is Markovian.

This argument may be generalized beyond reinforcement learning. Even when solving a task can in principle be based on direct input-output mapping, exploiting temporal aspects of the task may make learning easier. Indeed, it might in some cases yield more robust systems, for instance when the system’s inputs are very noisy so the pattern recognition process is unreliable. In the context of neural networks, it means that even if a feedforward network could be used to solve a task, it might sometimes be more efficient or effective to use recurrent neural networks.

Finally, as alluded to in the introduction, this type of research has some relevance for the debate in cognitive science regarding the status of representations in intelligent systems. Clark [4] argues that some form of representation may sometimes be necessary in an intelligent system to “stand in” for inputs that are either “absent” or “unruly”. In terms of Markovian and non-Markovian tasks, absent inputs correspond to a non-Markovian task. Unruly inputs are what this paper is concerned with. The task is Markovian, so in principle the best action can be derived from the input; but the input pattern is unruly, so the pattern recognition problem is hard. If, for this purpose, “some form of representation” is taken to mean “internal state”, this and other research of its kind provides “empirical” support for Clark’s “philosophical” argument. In absent cases, internal state is necessary by definition, and this paper shows that in unruly cases, internal state may be very useful.

References

- [1] A.G. Barto and R.S. Sutton. Landmark learning: An illustration of associative search. *Biological Cybernetics*, 42:1–8, 1981.
- [2] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.
- [3] A.R. Cassandra, L.P. Kaelbling, and M.L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, 1994.
- [4] A. Clark. *Being there: Putting mind, body, and world together again*. MIT Press (A Bradford Book), Cambridge, MA, 1997.
- [5] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
- [6] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
- [7] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning*, 8:293–321, 1992.
- [8] L.-J. Lin and T. Mitchell. Memory approaches to reinforcement learning in non-markovian domains. Technical report CMU-CS-92-138, Carnegie Mellon University, School of Computer Science, 1992.
- [9] L.-J. Lin and T. Mitchell. Reinforcement learning with hidden states. In J.-A. Meyer, H.L. Roitblat, and S.W. Wilson, editors, *From animals to animats 2: Proceedings of the second international conference on simulation of adaptive behavior*. MIT Press, Cambridge, MA, 1992.
- [10] R.A. McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 387–395. Morgan Kaufman, San Francisco, CA, 1995.
- [11] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel distributed processing: Explorations in the microstructure of cognition*, volume 1: Foundations. MIT Press (A Bradford Book), Cambridge, MA, 1986.
- [12] G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, 1994.
- [13] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.
- [14] R.S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems: Proceedings of the 1995 conference*, pages 1038–1044. Morgan Kaufman, San Francisco, 1996.
- [15] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT Press (A Bradford Book), Cambridge, MA, 1998.
- [16] C.J.C.H. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.