Reinforcement Learning with Perceptual Aliasing: The Perceptual Distinctions Approach

Lonnie Chrisman
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
chrisman@cs.cmu.edu

Abstract

It is known that *Perceptual Aliasing* may significantly diminish the effectiveness of reinforcement learning algorithms [Whitehead and Ballard, 1991]. Perceptual aliasing occurs when multiple situations that are indistinguishable from immediate perceptual input require different responses from the system. For example, if a robot can only see forward, yet the presence of a battery charger behind it determines whether or not it should backup, immediate perception alone is insufficient for determining the most appropriate action. It is problematic since reinforcement algorithms typically learn a control policy from immediate perceptual input to the optimal choice of action.

This paper introduces the *predictive distinctions approach* to compensate for perceptual aliasing caused from incomplete perception of the world. An additional component, a predictive model, is utilized to track aspects of the world that may not be visible at all times. In addition to the control policy, the model must also be learned, and to allow for stochastic actions and noisy perception, a probabilistic model is learned from experience. In the process, the system must discover, on its own, the important distinctions in the world. Experimental results are given for a simple simulated domain, and additional issues are discussed.

Introduction

Reinforcement learning techniques have recently received a lot of interest due to their potential application to the problem of learning situated behaviors for robotic tasks ([Sutton, 1990], [Lin, 1991], [Mahadevan and Connell, 1991], [Millán and Torras, 1991], [Chapman and Kaelbling, 1991]). The objective for a reinforcement learning agent is to acquire a policy for choosing actions so as to maximize overall performance. After each action, the environment provides feedback in the form of a scalar reinforcement value, and the *discounted cumulative reinforcement* is customarily used to assess overall performance.

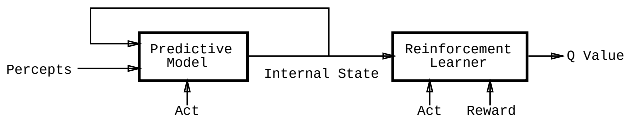

Figure 1: Data Flow Through System Components.

The effectiveness of reinforcement learning techniques may significantly diminish when there exist pertinent aspects of the world state that are not directly observable. The difficulty arises from what [Whitehead and Ballard, 1991] have termed *perceptual aliasing*, in which two or more perceptually identical states require different responses. An agent that learns its behavior as a function from immediate percepts to choice of action will be susceptible to perceptual aliasing effects. Nevertheless, common factors such as the presence of physical obstructions, limited sensing resources, and restricted field of view or resolution of actual sensors make incomplete observability a ubiquitous facet of robotic systems.

The Lion algorithm [Whitehead and Ballard, 1991], the CS-QL algorithm [Tan, 1991], and the INVOKE-N algorithm [Wixson, 1991] were previously introduced to cope with perceptual aliasing. Each of these algorithms compensates for aliasing effects by accessing additional immediate sensory input. This paper introduces a new approach that overcomes limitations of previous techniques in two important ways. First, assumptions of deterministic actions and noiseless sensing are dropped. And second, the new technique applies to tasks requiring memory as a result of *incomplete perception* [Chrisman *et al.*, 1991]. For example, if a warehouse robot has permanently closed and sealed a box and the box's contents determines its next action, it is necessary to remember the box's contents. Incomplete perception of this sort cannot be overcome by obtaining additional immediate perceptual input.

The current *predictive distinction approach* introduces an additional predictive model into the system[1],

[1]Predictive models have been used in reinforcement learning systems for various purposes such as experience

as shown in Figure 1. The predictive model tracks the world state, even though various features might not be visible at all times. Instead of learning a transfer function from percepts to evaluation of action, reinforcement learning now learns a transfer function from the internal state of the predictive model to action evaluation. Deterministic actions and noiseless sensing are *not* assumed; therefore, the predictive model is probabilistic[2]. A sufficient predictive model will usually not be supplied to the system *a priori*, so the model must be acquired or improved as part of the learning process. Learning the model involves not only estimating transition and observation probabilities, but also discovering what the states of the world actually are (c.f., [Drescher, 1991]). This is because perceptual discriminations can no longer be assumed to correspond directly with world states. With a noisy sensor, it may be possible to observe two or more different percepts from the same state, or perceptual incompleteness may cause identical percepts to register from distinct world states.

In our experiments, the agent begins initially with a small, randomly generated predictive model. The agent proceeds to execute actions in the world, performing a variation of Q-learning [Watkins, 1989] for action selection using the internal state of the predictive model as if it were perceptual input. After some experience has been gathered, this experience is used to improve the current predictive model. Using maximum likelihood estimation, probabilities are updated. Next, the program attempts to detect distinctions in the world that are missing from its current model. When the experience gives statistically significant evidence in support of a missing discrimination, a new distinction is introduced by recursively partitioning the internal state space of the model and readjusting probabilities. The system then cycles, using the new improved model to support Q-learning.

The next section introduces the general form of the predictive model and the Bayesian estimation procedure that uses it to update belief about the state of the world. The process of reinforcement learning when the model is provided is then discussed, followed by the model learning algorithm. Some empirical results are reviewed, and finally important issues and future research topics are listed.

The Predictive Model

A predictive model is a theory that can be used to make predictions about the effects of actions and about what the agent expects to perceive. In general, it may be necessary to maintain internal state in order to track aspects of the world that may occasionally become unobservable to the agent. For example, to predict what will be seen after turning around, a predictive model should remember the contents of that location the last time it was visible. Interestingly, the ability to predict is not the characteristic that makes predictive models useful for overcoming perceptual aliasing. Instead, it is the internal state that is formed and utilized to make predictions which is valuable to the reinforcement learner. The central idea behind the current approach is that the information needed to maximize predictiveness is usually the same information missing from perceptually aliased inputs.

Predictions need not be deterministic, and in fact in this context, deterministic models are inappropriate. The models here are stochastic. It is assumed that at any single instant t, the world state s_t is in exactly one of a finite number of classes, $class(s_t) \in \{1, 2, ..., n\}$, and class identity alone is sufficient to stochastically determine both perceptual response and action effects (i.e., the *Markov assumption*). A single class in the model may correspond to several possible world states. The agent has available to it a finite set of actions $\mathcal{A}$. For each action and pair of classes, $a^A_{i,j}$ specifies the probability that executing action A from class i will move the world into class j. The class of the world state is never directly observable — only probabilistic clues to its identity are available in the form of percepts. The perceptual model, $b_j(v)$, specifies the probability of observing v when the world state is in class j. Together, $a^A_{i,j}$ and $b_j(v)$ form the complete predictive model. For this paper, v is assumed to be a finite and nominal (unordered) variable. Predictive models of this form are commonly referred to as *Partially Observable Markov Decision Processes* ([Lovejoy, 1991], [Monahan, 1982]).

The actual class of the world at any instant is never directly observable, and as a result, it is in general not possible to determine the current class with absolute certainty. Instead, a belief is maintained in the form of a probability vector $\vec{\pi}(t) = \langle \pi_1(t), \pi_2(t), ..., \pi_n(t) \rangle$, where $\pi_i(t)$ is the believed probability that $class(s_t)$ is i at the current time t. Whenever an action is executed and a new observation obtained, Bayesian conditioning is used to update the belief vector as follows

$$\pi_j(t+1) = k \cdot b_j(v) \sum_i a^A_{i,j} \pi_i(t)$$

where A is the action executed, v is the sensed percept, and k is a normalizing constant chosen so that the components of $\vec{\pi}(t+1)$ sum to one.

Reinforcement Learning

Once a predictive model is available to the system, the task of the reinforcement learner is to learn the value of each action from each possible belief state. Specifically, the system must learn the function $Q(\vec{\pi}, A) : \Re^n \times \mathcal{A} \rightarrow \Re$, which returns the estimated cumulative

replay (e.g., [Lin, 1991], DYNA [Sutton, 1990])

[2]It may also be possible to apply recurrent neural networks to learn and use a predictive model in a similar fashion ([Jordan and Rumelhart, 1992], [Lin, personal communication]).

discounted rewards (referred to as *the Q value*) when given an action $A \in \mathcal{A}$ and a vector of probabilities $\vec{\pi}$ specifying the belief about the state of the world. To learn this function, a variation of the Q-learning algorithm ([Watkins, 1989], [Barto *et al.*, 1991]) is used. However, the Q-learning algorithm must be modified in this case because the domain of Q is not discrete and finite as the unmodified algorithm requires.

To learn the Q function, a very simple approximation is used. For each class i in the predictive model and each action A, a value $V[i, A]$ is learned. $Q(\vec{\pi}, A)$ is then approximated as

$$Q(\vec{\pi}, A) \approx \sum_{i=1}^{n} \pi_i V[i, A]$$

This approximation is accurate when the model is highly predictive of the class distinctions that impact the optimal control, so that most probability mass is usually distributed among classes that agree upon the optimal action. This approximation works well for many applications requiring memory to address perceptual aliasing, but is inappropriate for choosing between information gathering actions and active perception.

Learning Q involves only adjusting the values of V. This is done using the Q-learning rule, except that each class is treated as being only fractionally occupied. Each entry is adjusted by that fraction using the update rule

$$\begin{aligned} V[i, A] &= (1 - \beta\pi_i(t))V[i, A] + \\ &\quad \beta\pi_i(t)(r + \gamma U(\vec{\pi}(t+1))) \\ U(\vec{\pi}(t+1)) &= \max_A Q(\vec{\pi}(t+1), A) \end{aligned}$$

where A is the action taken, r is the reward received, γ is a discount factor, and β is the learning rate. All $V[i, A]$ for $i = 1, ..., n$ are updated after each action. If the model identifies a single class as the only one with a non-zero probability mass, the update rule reduces to conventional Q-learning.

With a predictive model in hand and the above updating rule in place, the overall scenario for this part of the system is the same as with most reinforcement learning systems. At each cycle, the agent obtains its current state estimate $\vec{\pi}(t)$ (in this case, from its predictive model) and executes the action having the largest $Q(\vec{\pi}(t), A)$. After the action completes, Bayesian conditioning uses $\pi(t)$ and the new perceptual input v to obtain $\vec{\pi}(t+1)$. The updating rule is applied and the cycle repeats.

Model Learning

Experience obtained by the agent from executing actions and the resulting perceptual input is used to improve its current predictive model. The task of the model learning algorithm is to obtain the best predictive model it can from this experience. This involves two aspects. First, given a set of classes, both the action transition probabilities $a_{i,j}^{A}$ and the observation probabilities $b_j(v)$ must be adjusted in order to maximize predictiveness. Second, the algorithm must detect and incorporate any additional distinctions existing in the world that are not currently accounted for by the model. Incorporating a new distinction involves enlarging the number of classes recognized by the model. If no initial predictive model is supplied to the system, the system begins with a two state model with randomized probabilities and then uses the algorithm to improve and enlarge it from experience.

Being probabilistic, model learning involves statistical assessment, making it is necessary to collect a body of experience before running the model learning algorithm. The agent executes for a prespecified number of cycles (m), recording each action-observation pair and continuously performing (modified) Q-learning. The agent then invokes the model learning algorithm, using the recorded experience as input, to produce an improved predictive model. The entire process then repeats. Since policy and model learning are interleaved, model learning is sensitive to the current control policy (c.f., [Jordan and Rumelhart, 1992]). As the control policy tends towards optimality, recorded experience will be primarily composed of states on the path to goal achievement, leading the model learning algorithm to learn mostly about situations that impact goal achievement.

Probability Adjustment

The first task in improving the model is adjusting the probabilities $a_{i,j}^{A}$ and $b_j(v)$ so as to maximize the predictiveness of the model. Simultaneously learning both action and perception models presents difficulties since the true class of the world is never directly revealed to the system. If the true class of the world were known at each instant, the problem would be trivial since the transition and observation *frequencies* could simply be counted and used. Nevertheless, it is possible to use the probability distribution $\vec{\pi}(t)$, representing the belief about the class of the world. For example, if at time t_1 the `DROP` action is executed, we can use $\vec{\pi}(t_1)$ and $\vec{\pi}(t_1 + 1)$ to assess the expected, rather than the actual, transition. If $\pi_2(t_1) = 0.6$ and $\pi_7(t_1 + 1) = 0.9$, then the count of the number of times that the `DROP` action results in a transition from class 2 to class 7 is incremented by $0.6 \times 0.9 = 0.54$. After the expected counts are tallied, the counts are divided as usual to obtain expected frequencies, which are then used for the resulting model probabilities. Before frequency counting, the Baum forward-backward procedure [Rabiner, 1989] is used to obtain improved an estimate for $\vec{\pi}$. The Baum forward-backward procedure is an efficient dynamic programming algorithm with a run-time complexity of $O(m \cdot |\lambda|)$, where m is the length of the experience trace, and $|\lambda|$ is size of the model (i.e., the number of probabilities in the model).

After the model (denoted by λ) has been adjusted using the above procedure, the process must be repeated until quiescence is reached. In the current system, quiescence is detected when no parameter of the model changes by more than 0.01. Let $A(t)$ denote the action taken at time t. [Baum *et al.*, 1970] proved that each iteration of this algorithm is guaranteed to improve the predictive power of the model, measured by $Pr(v(0),...,v(m)|A(1),...,A(m),\lambda)$, until quiescence is reached. The algorithm for this portion of the model learning is a variant of the "Balm-Welsh" algorithm for maximum likelihood estimation adapted here to learn Partially Observable Markov Decision Processes.

Discovering New Distinctions

It is generally not known beforehand how many classes suffice for obtaining the necessary level of predictability, or what these classes are. The second portion of the algorithm is responsible for detecting when the current model is missing important distinctions and for incorporating them into the model.

The primary challenge in discovering important distinctions is detecting the difference between random chance and underlying hidden influences missing from the current model. This is done in the system by performing two or more experiments under slightly different conditions and comparing the experiences. When the behavior of the system differs by a statistically significant amount between experiments, it is determined that an underlying influence is missing from the model (the unknown influence is at least partially determined by the experimental conditions), and so a new distinction is introduced. In the current case, this turns out to be equivalent to detecting when the Markov property of the model does not hold.

When the model learning algorithm is invoked, a sequential list of action-observation pairs has already been recorded and given to the algorithm as input. This experience is partioned into two groups with the earliest half forming the first group and the latest half forming the second group. Because reinforcement learning was actively changing the agent's policy while the experience was being gathered, the experimental conditions (determined by the policy) will be slightly different between the two groups. This forms the basis for detecting a missing distinction.

For each class i of the model, the expected frequencies from each group are tallied. For each action A and each class j, this yields the estimates ${}_1t^A_{i,j}$ and ${}_2t^A_{i,j}$ for the number of expected transitions from i to j with action A for groups 1 and 2 respectively. These two sampled distributions are compared for a statistically significant difference using the Chi-squared test. A similar test is performed for the observation counts ${}_1u_j(v)$ and ${}_2u_j(v)$, the expected number of times v is observed while in class j. If either test shows a statistically significant difference in distribution, class i is split into two, causing the total number of classes, n, to increment by 1. This is somewhat reminiscent of the G algorithm [Chapman and Kaelbling, 1991]. Whenever a class is split, the complete model learning algorithm is recursively re-invoked.

Experimental Results

This section reports the results of applying the complete system to a simple simulated docking application with incomplete perception, non-deterministic actions, and noisy sensors. The scenario consists of two space stations separated by a small amount of free space with loading docks located on each station. The task is to transport supplies between the two docks. Each time the agent successfully attaches to the least-recently visited station, it receives a reward of +10. In order to dock, the agent must position itself in front of the station, with its back to the dock, and backup. Whenever the agent collides with the station by propelling forward into the dock, it receives a penalty of -3. At all other times, it receives zero reinforcement.

Three actions are available to the agent: `GoForward`, `Backup`, `TurnAround`. The agent is always facing exactly one of the two stations, and `TurnAround` causes it to face the other. Depending on the current state, the `GoForward` action either detaches from the loading dock, launches into free space, approaches the next station from free space, or collides (with a penalty) into a space station directly ahead. `Backup` is almost the inverse of `GoForward` except that it is extremely unreliable. `Backup` launches from a station with probability 0.3. From space, it approaches a station in reverse with probability 0.8. And from docking position, it fails to dock 30% of the time. When actions fail, the agent sometimes remains in the same position, but sometimes accidentally gets randomly turned around.

The agent's perception is very limited. From a station, it sees the station or only empty space depending on which way it is facing. In free space perception is noisy: with probability 0.7 the agent sees the forward station, otherwise it sees nothing but empty space. The two stations appear identical to the agent except that the least-recently visited station displays an "accepting deliveries" sign which is visible to the agent exactly when the station is visible. When docked, only the interior of the dock itself is visible. The +10 reward is also observable for one time unit after receipt.

The system began with a randomly generated two state predictive model and zero-initialized Q values (i.e., $V[\cdot,\cdot] = 0$). Model learning was invoked after each $m = 1000$ actions to improve the model. The complete cycle was repeated 15 times. A discounting rate of $\gamma = 0.9$ and learning rate of $\beta = 0.1$ were used for reinforcement learning. To detect missing distinctions with the Chi-Squared test, a significance level of $\alpha = 0.05$ was used. Throughout the run, the agent executed a random action with probability 0.1, and executed the action with the largest Q the rest of the time.

In Figure 2, one dot was plotted corresponding to

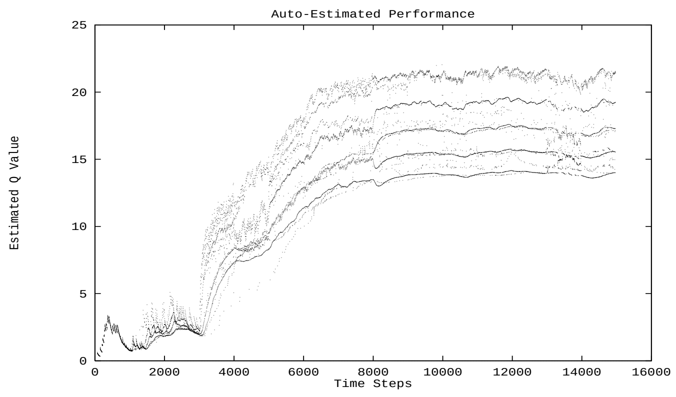

Figure 2: Estimated Performance during Training.

the agent's best estimate for the utility of the current choice of action at each decision cycle[3]. From the density of dots, five lines appear to eventually stand out. These correspond to the five states of the world where the agent spends most of its time once the optimal policy is learned. From a loading dock, the agent detaches (with `GoForward`), launches into free space (`GoForward`), approaches the next station (`GoForward`), turns around, and then backs up to attach to the dock. From the egocentric viewpoint of the agent, it then appears to be where it had started. By the end of the run, it recognizes these five situations, and the Q value estimate for each is fairly stable, forming the five discernible lines on the graph. Although these five are the most frequently visited states, by the end the system actually learns a predictive model with a total of 11 classes.

The final learned model was compared to the simulator's reality. In one case, the learned model distinguishes two different regions of free space, one where the station is visible, the other where it is not, and sets the transition probabilities of launching into each respective region of free space to 0.7 and 0.3. The model is exactly equivalent to the simulator but based upon a different ontology. Along the optimal path, the model is detailed and accurate, although there are two extra classes that are not necessary. Off the optimal path, all existing (and no spurious) distinctions are identified, but a few transition probabilities are in error (undoubtedly due to the lack of experience with those situations).

[3]Using the agent's own estimate of its performance can sometimes be a misleading indication of actual performance. By comparing a graph of measured cumulative discounted rewards to Figure 2, it was verified that Figure 2 does give a valid indication of actual performance, although Figure 2 does make the convergence rate appear somewhat worse than it actually is. However, Figure 2 provides far more information, both about actual performance and about the internal workings of the system.

Additional Issues

The system has been run on several additional simple simulated applications, and from this experience, a number of issues have been identified. On a few applications, the system performed poorly, leading to an investigation for the underlying reasons and the identification of the first few issues below. A few additional concerns are also listed. Dealing with all these limitations constitutes area for future research.

The Exacerbated Exploration Problem: The exploration/exploitation tradeoff is a known difficulty with reinforcement learning in general ([Kaelbling, 1990], [Thrun, 1992]); however, the problem is amplified tremendously by perceptual incompleteness. The additional aggravation stems from the fact that the state space structure is not provided to the system, but must instead be discovered. The result is that the agent sometimes cannot tell the difference between unexplored portions of the world and heavily explored portions of the world because until it has discovered the difference, the two areas look the same. This is an inherent problem accompanying incomplete perception and not unique to the current approach. Increasing the frequency of choosing random actions from 0.1 to 0.3 sometimes overcame this problem, but there is reason to believe that efficiently overcoming this problem in general may require the use of an external teacher (e.g., [Whitehead, 1991], [Lin, 1991]).

The Problem of Extended Concealment of Crucial Features: Some domains have the characteristic that some hidden feature or influence is crucial to performance, but the feature only rarely allows its influence to be perceived. This is perhaps the single most significant limitation to the predictive distinction approach. The problem is that high quality prediction is possible even when the crucial feature is ignored. In other words, the internal state that is useful for making predictions may, in some cases, not include the internal state necessary for selecting actions. This problem arises in the space station docking domain when the "accepting deliveries" sign is not used, leaving the agent the difficult task of discovering the crucial concept of "least-recently visited."

Oversplitting: It is common for the current system to learn more classes than are actually necessary. Conglomerating nearly identical states may be desirable (c.f., [Mahadevan and Connell, 1991]).

The Utile-Distinction Conjecture: Is it possible to only introduce those class distinctions that impact utility? If the color of a block is perceivable but irrelevant to the agent's task, is it possible for the agent to avoid introducing the color distinction into its model, while at the same time learning distinctions that are utile? I conjecture that this is not possible and that it is necessary to recognize a distinction and gather experience after the distinction is identified in order to obtain any information regarding the utility of the distinction. A refutation to this conjecture would be extremely interesting and would also provide an ideal solution to the input generalization problem ([Chapman and Kaelbling, 1991]).

Conclusion

Perceptual aliasing presents serious troubles for standard reinforcement learning algorithms. Standard algorithms may become unstable as the result of perceptually identical states that require different responses ([Whitehead and Ballard, 1991]). The *predictive distinction approach* uses a predictive model to track portions of the world that are not totally observable. It assumes an adequate model cannot be supplied to the system, so the model itself must be learned. The model is fully probabilistic and learning it involves not only learning transition and perception probabilities, but also discovering the important underlying class distinctions that exist in the world. Bayesian updating and conditioning track the world state, and a variation of Q-learning [Watkins, 1989] learns a mapping from the internal state of the model to the utility of each possible action. The overall approach is based upon the central idea that internal state useful for prediction may capture the important information for choosing actions missing from perceptually aliased inputs.

Acknowledgements

I am very grateful to Tom Mitchell for help and guidance during the development of this work, and to Tom Mitchell, Reid Simmons, and Sebastian Thrun for reading and making helpful comments on previous drafts. The research was sponsored by NASA under contract number NAGW-1175. The views and conclusions contained in this paper are those of the author and should not be interpreted as representing the official policies, either expresed or implied, of NASA or the U.S. government.

References

Barto, Andy G.; Bradtke, Steven J.; and Singh, Satinder P. 1991. Real-time learning of control using asynchronous dynamic programming. Technical Report COINS 91-57, Department of Computer Science, University of Massachusetts.

Baum, Leonard E.; Petrie, Ted; Soules, George; and Weiss, Norman 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathmatical Statistics* 41(1):164–171.

Chapman, David and Kaelbling, Leslie Pack 1991. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *IJCAI.*

Chrisman, Lonnie; Caruana, Rich; and Carriker, Wayne 1991. Intelligent agent design issues: Internal agent state and incomplete perception. In *Proc. AAAI Fall Symposium on Sensory Aspects of Robotic Intelligence.*

Drescher, Gary L. 1991. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence.* MIT Press.

Jordan, Michael I. and Rumelhart, David E. 1992. Forward models: Supervised learning with a distal teacher. *Cognitive Science.* In press. See also MIT Center for Cognitive Science Occasional Paper #40.

Kaelbling, Leslie Pack 1990. *Learning in Embedded Systems.* Ph.D. Dissertation, Stanford University. Teleos TR-90-04.

Lin, Long-Ji 1991. Programming robots using reinforcement learning and teaching. In *Proc. Ninth National Conference on Artificial Intelligence.*

Lin, Long-Ji 1992. personal communication.

Lovejoy, William S. 1991. A survey of algorithmic methods for partially observable markov decision processes. *Annals of Operations Research* 28:47–66.

Mahadevan, Sridhar and Connell, Jonathan 1991. Automatic programming of behavior-based robots using reinforcement learning. In *Proc. Ninth National Conference on Artificial Intelligence.*

Millán, José del R. and Torras, Carme 1991. Learning to avoid obstacles through reinforcement. In *Proc. Eighth International Machine Learning Workshop.*

Monahan, George E. 1982. A survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science* 28:1–16.

Rabiner, Lawrence R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2).

Sutton, Richard S. 1990. Integrated architecture for learning, planning, and reaction based on approximating dynamic programming. In *Proc. Seventh International Conference on Machine Learning.*

Tan, Ming 1991. Cost-sensitive reinforcement learning for adaptive classification and control. In *Proc. Ninth National Conference on Artificial Intelligence.*

Thrun, Sebastian B. 1992. Efficient exploration in reinforcement learning. Technical Report CS-CMU-92-102, School of Computer Science, Carnegie Mellon University.

Watkins, Chris 1989. *Learning from Delayed Rewards.* Ph.D. Dissertation, Cambridge University.

Whitehead, Steven D. and Ballard, Dana H. 1991. Learning to perceive and act by trial and error. *Machine Learning* 7:45–83.

Whitehead, Steven D. 1991. A complexity analysis of cooperative mechanisms in reinforcement learning. In *Proc. Ninth National Conference on Artificial Intelligence.*

Wixson, Lambert E. 1991. Scaling reinforcement learning techniques via modularity. In *Proc. Eighth International Machine Learning Workshop.*