UC Merced

Proceedings of the Annual Meeting of the Cognitive Science Society

Title

Cognitive Modeling of Action Selection Learning

Permalink

https://escholarship.org/uc/item/9t31r5j0

Journal

Proceedings of the Annual Meeting of the Cognitive Science Society, 18(0)

Authors

Gordon, Diana F.

Subramanian, Devika

Publication Date

1996

Peer reviewed

eScholarship.org Powered by the California Digital Library University of California

Cognitive Modeling of Action Selection Learning

Diana F. Gordon
Naval Research Laboratory, Code 5510
4555 Overlook Avenue, S.W.
Washington, D.C. 20375
gordon@aic.nrl.navy.mil

Devika Subramanian
Department of Computer Science
Rice University
Houston, TX 77005
devika@cs.rice.edu

Abstract

Our goal is to develop a hybrid cognitive model of how humans acquire skills on complex cognitive tasks. We are pursuing this goal by designing hybrid computational architectures for the NRL Navigation task, which requires competent sensorimotor coordination. In this paper, we describe results of directly fitting human execution data on this task. We next present and then empirically compare two methods for modeling control knowledge acquisition (reinforcement learning and a novel variant of action models) with human learning on the task. The paper concludes with an experimental demonstration of the impact of background knowledge on system performance. Our results indicate that the performance of our action models approach more closely approximates the rate of human learning on this task than does reinforcement learning.

Introduction

Our goal is to develop a hybrid cognitive model of how humans acquire skills by explicit instruction and repeated practice on complex cognitive tasks. We are pursuing this goal by designing hybrid computational architectures for the NRL Navigation task, which requires sensorimotor coordination skill. In this paper, we develop a novel method based on parametric action models for actively learning visual-motor coordination. Although similar to previous work on action models, our method is novel because it capitalizes on available background knowledge regarding sensor relevance. We have confirmed the existence and use of such knowledge with extensive verbal protocol data collected from human subjects. In our action models approach, the agent actively interacts with its environment by gathering *execution traces* (time-indexed streams of visual inputs and motor outputs) and by learning a compact representation of an effective policy for action choice guided by the action model.

This paper begins by describing the NRL Navigation task, as well as the types of data collected from human subjects performing the task. We next present the results of fitting the data directly. Then, two learning methods are described: our model-based method and a benchmark reinforcement learning algorithm that does not have an explicit model. Prior results reported in the literature of empirical comparisons of action models versus reinforcement learning are mixed (Lin, 1992; Mahadevan, 1992); they do not clearly indicate that one method is superior. Here we compare these two methods empirically on the Navigation task using a large collection of execution traces. Our primary goal in this comparison is to determine which performs more like human learning on this task. Both methods include sensor relevance knowledge from the verbal protocols. The results of this empirical comparison indicates that our action models method more closely approximates the time-scales and trends in human learning behavior on this task.

The NRL Navigation and Mine Avoidance Domain

The NRL navigation and mine avoidance domain, developed by Alan Schultz at the Naval Research Laboratory and hereafter abbreviated the "Navigation task," is a simulation that can be run either by humans through a graphical interface, or by an automated agent. The task involves learning to navigate through obstacles in a two-dimensional world. A single agent controls an autonomous underwater vehicle (AUV) that has to avoid mines and rendezvous with a stationary target before exhausting its fuel. The mines may be stationary, drifting, or seeking. Time is divided into episodes. An episode begins with the agent on one side of the mine field, the target placed randomly on the other side of the mine field, and random mine locations within a bounded region. An episode ends with one of three possible outcomes: the agent reaches the goal (success), hits a mine (failure), or exhausts its fuel (failure). Reinforcement, in the form of a binary reward dependent on the outcome, is received at the end of each episode. An episode is further subdivided into decision cycles corresponding to actions (decisions) taken by the agent.

The agent has a limited capacity to observe the world it is in; in particular, it obtains information about its proximal environs through a set of seven consecutive sonar segments that give it a 90 degree forward field of view for a short distance. Obstacles in the field of view cause a reduction in sonar segment length; one mine may appear in multiple segments. The agent also has a range sensor that provides the current distance to the target, a bearing sensor that indicates the direction in which the target lies, and a time sensor that measures the remaining fuel. A human subject performing this task sees visual gauges corresponding to each of these sensors. The turn and speed actions are controlled by joystick motions. The turn and speed chosen on the *previous* decision cycle are additionally available to the agent. Given its delayed reward structure and the fact that the world is presented to the agent via sensors that are inadequate to guarantee correct identification of the current state, the Navigation world is a partially observable Markov decision process (POMDP).

Data from Human Subjects

In the experiments with humans, seven subjects were used, and each ran for two or three 45-minute sessions with the simulations. We instrumented[1] the simulation to gather execution traces for subsequent analysis (Gordon *et al.*, 1994). We also obtained verbal protocols by recording subject utterances during play and by collecting answers to questions posed at the end of the individual sessions.

Fitting the Human Data

Our first task was to directly fit the execution trace data from a human subject who had become an expert at the task. In other words, our goal was to learn a stimulus-response controller that represents the expert policy used by the subject. This control policy can be expressed as the function F, where:

$$F : \text{sensors} \rightarrow \text{actions}$$

The task was configured with a small (5%) amount of sensor noise and 25 stationary mines. We used 312 time-indexed execution trace snapshots collected from an expert subject. Each snapshot had the sensor values plus the corresponding action taken by the subject under these conditions. Four supervised inductive learning paradigms induced controllers from this data: CART (Breiman *et al.*, 1984), C4.5 (Quinlan, 1986), MDL (Rissanen, 1983), and backpropagation in neural networks (Rumelhart & McClelland, 1986). For all of these methods the 12 inputs were the sensor values plus the value of the last turn and last speed, and the output was an action chosen by our subject. One decision tree or neural net was constructed for predicting the subject's next turn. A comparison of the accuracies of the four methods on this prediction task is shown in Table 1. The neural network function fitter had the smallest mean-squared error. We experimented with a wide range of parameters for the neural networks and report the results with the best settings (sweeps = 10^6, 9 hidden units, learning rate $\alpha = 0.1$). Although the mean-squared error of fit for the symbolic learning methods was higher, the structures produced by them (especially C4.5) revealed interesting aspects of the control strategy used by the subject

Method	Accuracy	MSE	Rep. Complexity
CART	85.3	0.25	5 leaves in tree
C4.5	87.8	0.22	10 leaves in tree
MDL	82.05	0.27	17 leaves in tree
NN	92.0	0.11	9 hidden units

Table 1: Results of the construction of stimulus-response controllers from execution trace data collected from human subjects.

that were remarkably consistent with the subject's verbal protocol data. For instance, it demonstrated that the subjects did not give equal importance to all the sonar information. In addition, the last turn action played a crucial role in determining current turn. The insights about the differential relevance of various pieces of information, which was obtained by examining these decision trees inspired us to continue working with C4.5 for modeling action choice (see below).

Methods for Modeling Action Selection Learning

After having fit the data from a human expert at the task, our next goal is to build a model that most closely duplicates the human subject data in *learning* performance, i.e., in transitioning from a novice to an expert. With no sensor noise and only 25 mines, all of our subjects became experts at this task after only a few episodes. Modeling such an extremely rapid learning rate presents a challenge. In developing our learning methods, we have drawn from both the machine learning and cognitive science literature. By far the most widely used machine learning method for tasks like ours is reinforcement learning. Reinforcement learning is mathematically sufficient for learning policies for our task, yet has no explicit world model. More common in the cognitive science literature are action models, e.g., (Arbib, 1972), which require building explicit representations of the dynamics of the world to choose actions.

Reinforcement learning

Reinforcement learning has been studied extensively in the psychological literature, e.g., (Skinner, 1984), and has recently become very popular in the machine learning literature, e.g., (Sutton, 1988; Lin, 1992; Gordon & Subramanian, 1993). Rather than using only the difference between the prediction and the true reward for the error, as in traditional supervised learning, (temporal difference) reinforcement learning methods use the difference between successive predictions for errors to improve the learning. Reinforcement learning provides a method for modeling the acquisition of the function F, described above.

Currently, the most popular type of reinforcement learning is *q-learning*, developed by Watkins, which is based on ideas from temporal difference learning, as well as conventional dynamic programming (Watkins, 1989). It requires estimating the q-value of a sensor configuration s, i.e., $q(s, a)$ is a prediction of the utility of taking

[1]Note that although human subjects use a joystick for actions, we do not model the joystick but instead model actions at the level of discrete turns and speeds (e.g., turn 32 degrees to the left at speed 20). Human joystick motions are ultimately translated to these turn and speed values before being passed to the simulated task. Likewise, the learning agents we construct do not "see" gauges but instead get the numeric sensor values directly from the simulation (e.g., range is 500).

action a in a world state represented by s. The q-values are updated during learning based on minimizing a temporal difference error. Action choice is typically stochastic, where a higher q-value implies a higher probability that action will be chosen in that state.

While q-learning with explicit state representations addresses the temporal credit assignment problem, it is standard practice to use input generalization and neural networks to also address the structural credit assignment problem, e.g., (Lin, 1992). The q-value output node of the control neural network corresponding to the chosen action a is given an error that reflects the difference between the current prediction of the utility, $q(s_1, a_i)$, and a better estimate of the utility (using the reward) of what this prediction should be:

$$error_i =$$

$$\begin{cases} (r + \gamma\ max\{q(s_2, k) | k \in A\}) - q(s_1, a_i) & \text{if } a_i = a \\ 0 & \text{otherwise} \end{cases}$$

where r is the reward, A is the set of available actions, a is the chosen action, s_2 is the state achieved by performing action a in state s_1, i indexes the possible actions, and $0 \leq \gamma < 1$ is a discount factor that controls the learning rate. This error is used to update the neural network weights using standard backpropagation. The result is improved q-values at the output nodes.

We selected q-learning as a benchmark algorithm with which to compare because the literature reports a wide range of successes with this algorithm, including on tasks with aspects similar to the NRL Navigation task, e.g., see (Lin, 1992). Our implementation uses standard q-learning with neural networks. One network corresponds to each action (i.e., there are three turn networks corresponding to turn left, turn right, and go straight; speed is fixed at a level frequently found in the human execution traces, i.e., 20/40). Each turn network has one input node for every one of the 12 sensor inputs (e.g., one for bearing, one for each sonar segment, etc.), one hidden layer[2] consisting of 10 hidden units, and a single output node corresponding to the q-value for that action. A Boltzmann distribution is used to stochastically make the final turn choice:

$$probability(a|s) = e^{q(s,a)/T} / \sum_i e^{q(s,a_i)/T} \quad (1)$$

where s is a state and the temperature T controls the degree of randomness of action choice.

We use a reward r composed of a weighted sum of the sensor values.[3] Our reward models sensor relevance information derived from the human subjects data we collected. These subjects appeared to learn relevance knowledge and action selection knowledge simultaneously. Here, we assume the relevance is known. Future work will involve methods for acquiring relevance knowledge.

The verbal protocols from human subjects reveal that the sonar and bearing sensors appear to be critical for action selection. Furthermore, the middle three sonar segments (which show what is directly ahead) appear to be the most critical of the sonar segments. This is logical: after all, the middle three sonar segments show a mine straight ahead so you can avoid collisions, and the bearing tells you whether you are navigating toward or away from the target. Based on the verbal protocol data, we have implemented a reward function that weights the bearing equally to the three sonar segments and gives other sensors zero weight. Thus, if the bearing shows the target straight ahead and the middle three sonar segments show no obstacles, then the reward is highest.

The verbal protocols also indicate heuristics for focusing attention on different sensors at different times. This knowledge is implemented in our novel variant of action models, described next. Nevertheless it is not implemented in the q-learner because to do so would require a departure from the standard q-learning architecture reported in the literature, with which we wish to compare as a benchmark.

Learning action models

One of the more striking aspects of the verbal protocols we collected was that subjects exhibited a tendency to build internal models of actions and their consequences, i.e., *forward models* of the world. These expectations produced surprise, disappointment, or positive reinforcement, depending on whether or not the predictions matched the actual results of performing the action. For example, one subject had an expectation of the results of a certain joystick motion: "Why am I turning to the left when I don't feel like I am moving the joystick much to the left?" Another expressed surprise: "It feels strange to hit the target when the bearing is not directly ahead." Yet a third subject developed a specific model of the consequences of his movements: "One small movement right or left seems to jump you over one box to the right or left," where each box refers to a visual depiction of a single sonar segment in the graphical interface.

Action models (i.e., forward models) have appeared in multidisciplinary sources in the literature. Arbib (1972) and Drescher (1991) provide examples in the psychological literature, STRIPS (Nilsson, 1980) is a classic example in the AI literature, and Sutton uses them in DYNA (Sutton, 1988). The *learning* of action models has been studied in the neural networks (Moore, 1992), machine learning (Sutton, 1990; Mahadevan, 1992), and cognitive science (Munro, 1987; Jordan & Rumelhart, 1992) communities.

[2]We ran initial experiments to try to optimize the reinforcement learning parameters. For the neural networks, the chosen learning rate is 0.5, momentum 0.1, 10 hidden units, and 10 training iterations for the neural networks and a discount factor of 0.9.

[3]Ron Sun suggested a reward of sensor values for this task (personal communication). Our choice of sensor weights for the reward is 30 for bearing and 10 for each of the three middle sonar segments, and the scale for the reward is between -1.0 and 0.

Our algorithm uses two functions:

$$A : \text{sensors} \times \text{actions} \rightarrow \text{sensors}$$
$$P : \text{sensors} \rightarrow \Re$$

A is an action model, which our method represents as a decision tree. The decision trees are learned using Quinlan's C4.5 system (Quinlan, 1986).[4] P rates the desirability of various sensor configurations. P embodies background (relevance) knowledge about the task. For sonars, high utilities are associated with large values (no or distant mines), and for the bearing sensor high utilities are associated with values closer to the target being straight ahead. Currently, P is supplied by us. At each time step, actions are selected using P and A by performing a 1-step lookahead with model A and rating sensory configurations generated using P. The action models algorithm has the same action set as the q-learning algorithm, i.e., turn right, turn left, or go straight at a fixed speed (20/40).

First, our algorithm goes through a training phase, during which random turns are taken and the execution traces saved as input for C4.5. C4.5 models the learning of the function A. In particular, it constructs two decision trees from the data: one tree to predict (from (s, a)) the *next* composite value of the middle three sonar segments (prediction choices are no-mines, mine-far, mine-mid, mine-fairly-close, or mine-close, where these nominal values are translations from the numeric sonar readings) and one tree to predict the bearing on the next time step. Note that the choice of these two trees employs the same relevance information used in the reinforcement learning reward function, namely, that the middle three sonar segments and bearing are the relevant sensors. The training phase concludes after C4.5 constructs these two decision trees.

During the testing phase, these trees representing the world dynamics (A) are consulted to make predictions and select turns. Given the current state, a tree is chosen. The tree selection heuristic for focus of attention states: if the middle three sonar segments are below a certain empirically determined threshold (150/220), the sonar prediction tree selects the *next* turn. Otherwise, the bearing prediction tree selects the next turn. To make a prediction, the agent feeds the current sensor readings (which include the *last* turn and speed) and a candidate *next* turn to the decision tree and the tree returns the predicted sonar or bearing value. The agent chooses the next turn which maximizes P.[5]

It is unlikely that humans recompute the consequences of actions when the current state is similar to one seen in the past. Therefore, our future work will focus on memorizing cases of successful action model use so that memory can be invoked, rather than the trees, for some predictions.

Empirical Evaluation of the Methods

To make our comparisons fair, we include a training phase for the reinforcement learner with Boltzmann temperature at 0.5, which results in random actions.[6] A testing phase follows in which the turn with the best q-value is selected deterministically at each time step. In summary, the reinforcement learner takes random actions and learns its q-values during training.[7] It uses these learned q-values for action selection during testing. The action models method takes the same random actions as the q-learner during training (i.e., it experiences exactly the same sensor and action training data as the q-learner), and then from this training data it learns decision tree action models. A heuristic uses the learned trees for action selection during testing. Neither of the two methods learns during testing. Both methods have the same knowledge regarding which sensors are relevant.

We denote the q-learning scheme described above as Q_{rel} and the action model scheme with decision trees described above as A_{rel}. Two schemes in which we have removed sensor relevance knowledge are denoted Q_{remrel} and A_{remrel} respectively and are described below.

We empirically test the following hypotheses:

- *Hypothesis 1:* The slope of A_{rel}'s learning curve is closer than Q_{rel}'s to the slope of the human learning curve, for the Navigation task.
- *Hypothesis 2:* The slope of A_{remrel}'s learning curve (respectively, Q_{remrel}) is lower than that of A_{rel} (respectively, Q_{rel}), for the Navigation task.

The justification for Hypothesis 1 is that our action models method uses an action choice policy specially designed to capitalize on sensor relevance knowledge. The justification for Hypothesis 2 is that removal of relevance knowledge should degrade performance.

In our experimental tests of these hypotheses, the training phase length is varied methodically at 25, 50, 75, and 100 episodes. The testing phase remains fixed at 400 episodes.[8] Each episode can last a maximum of 200 time steps, i.e., decision cycles. In all experiments, the number of mines is fixed at 25, there is a small amount of mine drift, and no sensor noise. These task parameter settings match exactly those used for the human subject whose learning we wish to model.[9] Performance is averaged over 10 experiments because the algorithms

[4]We are not claiming humans use decision trees for action models; however, we use this implementation because it appears to have a computational speed that is needed for modeling human learning. We are also investigating connectionist models as in Jordan & Rumelhart (1992). Currently, C4.5 learning is in batch. To more faithfully model human learning, we are planning to use an incremental version of decision tree learning in future implementations.

[5]If the next turn is considered irrelevant by the decision tree, a random action choice is made.

[6]We also tried an annealing schedule but performance did not improve.

[7]Arbib (1972) provides convincing cognitive justification for the role of random exploration of actions in the acquisition of motor skill.

[8]We experimented with the number of episodes and chose a setting where performance improvement leveled off for both algorithms.

[9]Both algorithms go straight (0 turn) for the first three time steps of every episode during training. This not only matches performance we observed in the execution traces

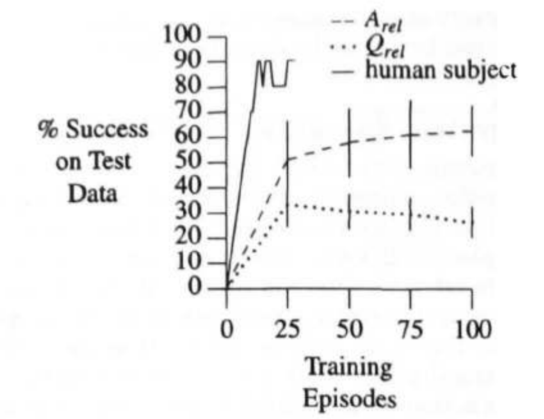

Figure 1: The graph represents the learning curves for A_{rel} and Q_{rel}. Note the learning curve of a human subject superimposed on this graph.

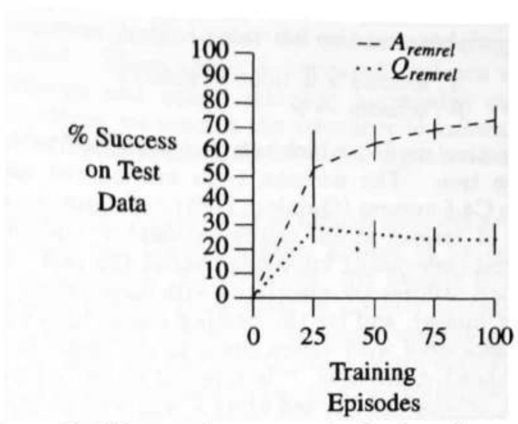

Figure 2: The graph represents the learning curves for A_{remrel} and Q_{remrel}.

are stochastic during training, and testing results depend upon the data seen during training. Graphs show mean performance, and error bars denote the standard deviation.

To test Hypothesis 1, we used data from a typical (the variance between subjects was surprisingly low) subject for a single 45-minute session. Note that we cannot divide the human learning into a training phase and a testing phase during which the human stops learning. Therefore, we have averaged performance over a sliding window of 10 previous episodes. We considered averaging performance over multiple subjects, but that would entail significant information loss.

Figure 1 shows the results of testing Hypothesis 1. The action models method outperforms reinforcement learning at a statistically significant level (using a paired, two-tailed t-test with $\alpha = 0.05$). Thus, Hypothesis 1 is confirmed.[10] Apparently, our novel method for coupling action models with an action choice policy that exploits sensor relevance has tremendous value for this task. We believe that one of the chief reasons for the superior performance of the action models method is the action choice heuristic for deciding when to pay attention to which sensor. Another likely reason for the dominance of action model-based methods is that neural networks with backpropagation tend to learn slowly (Chapman & Kaelbling, 1991).

To test Hypothesis 2, we made all seven, rather than just the three middle, sonar segments relevant for both algorithms, thereby creating A_{remrel} and Q_{remrel}. Figure 2 shows the results of testing Hypothesis 2. Q_{remrel} performs slightly worse than Q_{rel}. The performance drop is not statistically significant ($\alpha = 0.10$) for training lengths of 25, 50, and 100, but is significant ($\alpha = 0.10$) with a training length of 75. The surprise is that A_{remrel} outperforms A_{rel}. The differences are statistically significant at $\alpha = 0.05$ for training lengths of 75 and 100, but only at $\alpha = 0.20$ for training lengths of 25 and 50. Our results refute Hypothesis 2. Apparently, both methods are knowledge sensitive, though action models is more sensitive. We conjecture that the reason the action models method improves with more sensory information is that the extra sonar segments carry some useful information. The action models heuristic, which separates sonar and bearing information, is able to take advantage of the added sonar information. Because the q-learning method lumps all knowledge into one reward it probably needs longer training to do likewise. Further experiments are needed to test this hypothesis. (Although these results with A_{remrel} and Q_{remrel} refute Hypothesis 2, they provide further confirmation of Hypothesis 1 because the performance improvement of A_{remrel} over that of Q_{remrel} is statistically significant with $\alpha = 0.05$).

Discussion and Future Work

The most immediate pressing question is: why are *both* methods slower learners than the human? We believe humans have more knowledge when they begin this task, perhaps from driving or walking experience. We are presently working to make these forms of knowledge explicit, so we can more closely match human performance using our computational models. Another issue

from human subjects, but also aids the learning process by quickly moving the AUV into the mine field.

[10]It is unclear why the performance of the q-learner drops slightly with more training episodes, though perhaps overfitting explains this.

that we are currently investigating is whether the power law of practice holds for human subjects in this task, and whether our models conform to it. Performance levels off rather quickly both for humans and our models, making this determination challenging.

Future work will also focus on studies to determine the source of power of our action models approach over the *q*-learner for this task. As mentioned above, strong candidate explanations are (1) the focus of attention knowledge in the action models heuristic, (2) the use of an action model *per se*, and (3) the decision tree representation.

Acknowledgements

This research was sponsored by the Office of Naval Research N00014-95-WX30360 and N00014-95-1-0846. Special thanks to Sandy Marshall and William Spears for their helpful comments and suggestions, to Helen Gigley and Susan Chipman for their encouragement and support, to Jim Ballas for the joystick interface, and to Alan Schultz for all the work he did to tailor the simulation to both machine learning and human experimental needs.

References

Arbib, M.A. (1972). *The Metaphorical Brain.* NY: Wiley and Sons Publishers.

Breiman, L., Friedman, J.H. Olshen, R.A.. & Stone, C.J. (1984). *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group Publishers.

Chapman, D. & Kaelbling, L. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of Twelfth International Joint Conference on Artificial Intelligence* (pp. 726–731). San Mateo, CA: Morgan Kaufmann Publishers.

Drescher, G.L. (1991). *Made-Up Minds.* Canbridge, MA: MIT Press.

Gordon, D., Schultz, A., Grefenstette, J., Ballas, J., & Perez, M. (1994). *User's Guide to the Navigation and Collision Avoidance Task.* Naval Research Laboratory Technical Report AIC-94-013.

Gordon, D. & Subramanian, D. (1993). A Multistrategy Learning Scheme for Agent Knowledge Acquisition. *Informatica*, 17, 331–346.

Jordan, M.I. & Rumelhart, D.E. (1992). Forward models: supervised learning with a distal teacher. *Cognitive Science*, 16, 307–354.

Lin, L. (1992). Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8, 293–321.

Mahadevan, S. (1992). Enhancing transfer in reinforcement learning by building stochastic models of robot actions. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 290–299). San Mateo, CA: Morgan Kaufmann Publishers.

Moore, A. (1992). Fast, robust adaptive control by learning only forward models. In *Proceedings of the International Joint Conference on Neural Networks*, 4, (pp. 571–578). San Mateo, CA: Morgan Kaufmann.

Munro, P. (1987). A dual back-propagation scheme for scalar reward learning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 165–176). Hillsdale, NJ: Erlbaum.

Nilsson, N. (1980). *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga Publishing Company.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–107.

Rissanen, J. (1983). *Minimum Description Length Principle.* Report RJ 4131 (45769), IBM Research Laboratory, San Jose.

Rumelhart, D.E. & McClelland, J.L. (1986). *Parallel Distributed Processing : Explorations in the Microstructure of Cognition.* Cambridge, MA: MIT Press.

Skinner, B.F. (1984). Selection by consequences. *The Behavior and Brain Sciences*, 7, 477–510.

Sutton, R. (1988). Learning to Predict by the Methods of Temporal Differences. Machine Learning, 3, 9–44.

Sutton, R. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 216-224). San Mateo, CA: Morgan Kaufmann.

Watkins, C. (1989). Learning from Delayed Rewards. Doctoral dissertation. Cambridge, England: Cambridge University.