A Comparison of Action Selection Learning Methods

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 empirically compare two methods for control knowledge acquisition (reinforcement learning and a novel variant of action models), as well as a hybrid of these methods, with human learning on this task. Our results indicate that the performance of our action models approach more closely approximates the rate of human learning on the task than does reinforcement learning or the hybrid. We also experimentally explore the impact of background knowledge on system performance. By adding knowledge used by the action models system to the benchmark reinforcement learner, we elevate its performance above that of the action models system.

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 (multistrategy) 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. 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 indicate that our action models method more closely approximates the time-scales and trends in human learning behavior on this task. Nevertheless, neither algorithm performs as well as the human subject.

We next explore a multistrategy variant that combines the two methods for the purpose of better approximating the human learning, and present empirical results with this method. Although the multistrategy approach is unsuccessful, an alternative is highly successful. This alternative consists of modifying the architecture of the reinforcement learner to incorporate knowledge used by the action models method.

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 (segment length is proportional to obstacle distance); 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 (in clock notation) 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).

An example of a few snapshots from an execution trace (with only a subset of the sensors shown) is the following:

time	range	bearing	sonar1	turn	speed
4	1000	1	220	32	20
5	1000	12	220	-32	20
6	1000	11	220	0	20
7	1000	11	90	0	20

A trace file records the binary success/failure for each episode.

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.

[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).

Methods for Modeling Action Selection Learning

Our goal is to build a model that most closely duplicates the human subject data in learning performance. In particular, subjects become proficient at this task (assuming no noise in the sensors and only 25 mines) 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 policy function:

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

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 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:

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

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 a special type of sensor relevance information derived from the human subject data we collected — it represents knowledge of the relative importance of the various sensory inputs in determining action. The verbal protocols reveal that the sonar and bearing sensors appear to be critical for action selection. This is logical: after all, the sonar shows mines which you need to avoid, and the bearing tells you whether you are navigating toward or away from the target. We have implemented a reward function that weights the bearing and sonar equally and gives 0 weight to the other sensors. Thus, if the bearing shows the target straight ahead and the sonar segments show no obstacles, then the reward is highest. Our 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 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 initially as a benchmark. Later, we describe modifications to the q-learner to include this focusing knowledge.

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.

Our algorithm uses two functions:

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

M 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

[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 seven sonar segments, and the scale for the reward is between -1.0 and 0.

[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

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 M by performing a 1-step lookahead with model M 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 M. In particular, it constructs two decision trees from the data: one tree to predict (from (s, a)) the *next* composite value of the 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 sonar 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 (M) are consulted to make predictions and select turns. Given the current state, a tree is chosen. The tree selection heuristic for focus of attention (hereafter called the *focus heuristic*) states: if all the 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 address memorizing cases of successful action model use so that memory can be invoked, rather than the trees, for some predictions.

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.

Empirical Comparison of the Two 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 the execution trace training data it learns decision tree action models. The focus 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, i.e., the bearing and sonar sensors.

In our experimental tests of all 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]

Recall that the outcome of every episode is binary (success/failure) and that success implies the AUV avoids all the mines and reaches the target location. The performance measure we use is the percentage of test episodes in which the AUV succeeds. This information is obtained from the trace file. Performance is averaged over 10 experiments because the algorithms 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.

We denote the q-learning scheme described above as Q and the action model scheme with decision trees described above as A. Our goal is to find an algorithm that most closely approximates the human learning. We start with the basic algorithms (Q and A), then

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

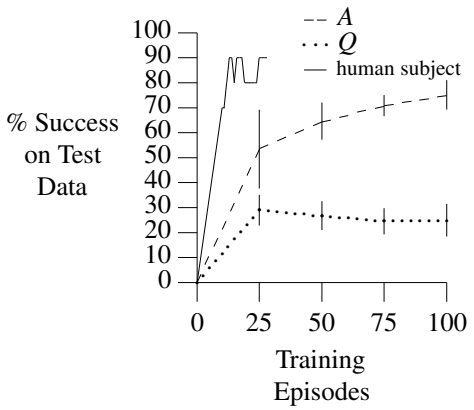

Figure 1: The graph represents the learning curves for A, Q, and the human subject.

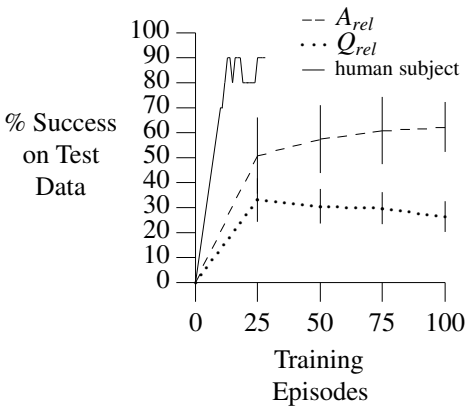

Figure 2: The graph represents the learning curves for A_{rel}, Q_{rel}, and the human subject.

make changes as needed to more closely approximate the human learning.

We begin by empirically testing the following hypothesis:

- *Hypothesis 1:* The slope of A's learning curve is closer than Q's to the slope of the human learning curve, for the Navigation task.

The justification for Hypothesis 1 is that our action models method uses an action choice policy, including a focus heuristic, specially designed to capitalize on sensor relevance knowledge.

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. A outperforms Q 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.

Among the most plausible explanations for the power of our action models approach over the q-learner for this task are: (1) A's focus of attention heuristic, (2) use of an action model *per se*, and (3) the decision tree representation, e.g., Chapman and Kaelbling (1991) discuss how decision trees can improve over neural networks with backpropagation for reinforcement learning.

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

Although A performs more like the human than Q, the most pressing question is why neither performs as well the human. We next add more knowledge in order to gain a better approximation of the curve of the human learner. A more careful examination of the verbal protocols indicates that subjects not only attended to the sonar sensor, but some subjects mentioned focusing almost exclusively on the middle three sonar segments. This makes sense if you consider the middle three sonar segments show mines straight ahead, which is critical information for avoiding immediate collisions. We have have altered Q's reward function to weight the bearing equally to the middle three sonar segments and to give all other sensors zero weight. Thus, if the bearing shows the target straight ahead and the middle three sonar segments show no obstacles ahead, then the reward is highest. A was also modified to include this more specific relevance knowledge, i.e., the modified version only predicts the values of the middle three sonar segments rather than all, and the focus heuristic checks the threshold for only those three segments. We denote the modified q-learning scheme Q_{rel} and the modified action model scheme A_{rel}.

We empirically test the following hypothesis:

- *Hypothesis 2:* The slope of A_{rel}'s learning curve (respectively, Q_{rel}) is higher than that of A (respectively, Q), for the Navigation task.

The justification for Hypothesis 2 is that the addition of relevance knowledge should improve performance.

Figure 2 shows the results of testing Hypothesis 2. If we compare Figure 2 with Figure 1, we see that Q_{rel} performs slightly better than Q. The performance difference 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_{rel} performs worse than A. 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.

(Although these results refute Hypothesis 2, they provide further confirmation of Hypothesis 1 because the performance improvement of A_{rel} over that of Q_{rel} is statistically significant with $\alpha = 0.05$).

Apparently, adding this relevance knowledge does not help A, which is the better performer. We conjecture the reason is that although the middle three sonar segments may be the most relevant, all the sonar segments are in fact relevant for action choice. For example, if there is a mine ahead as well as one to the left, then the best action to take to avoid the mines is to turn right. With knowledge only of what is straight ahead, turning right or left would seem equally valid. The subjects were most likely using this knowledge, but their verbal recollections were probably incomplete accounts of the full knowledge that they actually used for decision-making.

A Multistrategy Approach

The addition of knowledge about which sensors are relevant does not improve our algorithms. Therefore, we are still left with the question: why are *both* methods slower learners than the human?[11] While Q in principle could match A's performance, the search that it embodies has to be rescued from the performance plateau we find in Figures 1 and 2, in order to match human learning rates. Q possesses the ability to implicitly acquire the function P as well as the focus heuristic from task interaction. Therefore we examine the question: is it possible to combine algorithms A and Q to create a hybrid method that outperforms both, and which more closely matches the human learning curve on this task?

We are in the preliminary stages of investigating this question. The idea is to use A's superior performance after the same level of training to get Q out of its performance plateau and then use Q to refine the knowledge it acquired from A's focus heuristic and sensor evaluation function P. We have implemented the first part of our hybrid scheme in the algorithm MSL, described below.

MSL performs a random walk in the space of actions for the first 25 episodes, i.e., it experiences exactly the same training data as A and Q for the first 25 episodes. The A component of MSL then builds the decision trees. Next, MSL uses the focus heuristic with the learned decision trees to select actions from episodes 25 to 100. Concurrently, action choices and rewards are used to refine the q-values for Q. Since we have empirically established that the action choices of A are significantly better than those of Q after 25 episodes, we expect algorithm Q to find itself in a better region of the policy space during learning episodes 25 to 100. After the 100th episode, we turn off learning and test MSL for 400 episodes. In the testing phase, MSL chooses actions based on the learned q-values alone. The results of this experiment (shown below) are quite surprising. The performance of MSL turns out to be not statistically significantly different from that of Q ($\alpha = 0.20$). This experiment highlights some of the difficulties in constructing hybrid algorithms for complex tasks.

Algorithm	Training Length	Mean	Std
Q	100	24.92	6.62
MSL	100	27.07	8.56
A	25	53.50	15.81

It seems the multistrategy approach is not rescuing Q to make the combined system perform more like the human. In the next section, we try an alternative tactic, namely, that of adding what we consider to be one of the most useful elements of A into Q. Perhaps by adding elements of A one by one into Q we can eventually develop a system whose learning rate more closely approximates that of the human than either A or Q because it would include the best elements of both systems.

Adding the Focus Heuristic to Q

As mentioned above, the focus heuristic, the use of action models, and the use of decision trees are all candidate explanations for A's superior performance over that of Q. We begin by considering the impact of the focus heuristic. This heuristic separates sonar and bearing information and therefore is able to take advantage of knowledge regarding which sensors are relevant *when*. Because the q-learning method lumps all knowledge into one reward it probably needs longer training to do likewise. Here, we explore the impact of adding the focus heuristic to Q. Future work will investigate adding other elements of A to Q one by one.

A new version of Q, called Q_{focus}, has been created by generating two sets of neural networks, one for selecting the actions for making choices to improve the bearing and another set for improving the sonar. The first set receives the bearing component of the reward and the second receives the sonar component (i.e., these are the respective elements of the previous reward, which is a weighted sum). Arbitration between the two sets of networks is done by the same focus heuristic used by the action models approach, namely, use the sonar networks to select a turn (based on Boltzmann selection using the q-values) if the sonar values are below the threshold; otherwise use the bearing networks to select.

We empirically test the following hypothesis:

- *Hypothesis 3:* The slope of Q_{focus}'s learning curve is closer to that of the human's than Q's is.

[11] It may also be the case that their asymptotic performance is lower than that of humans, although A does show signs of a steady increase and the slope is positive even at 100 training episodes.

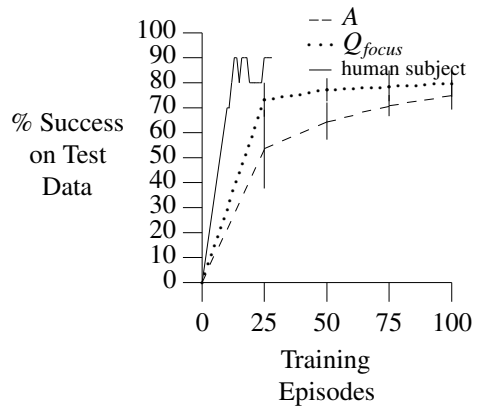

Figure 3: The graph represents the learning curves for A, Q_{focus}, and the human subject.

The results of testing Hypothesis 3 are in Figure 3. The hypothesis is confirmed, but what is unexpected is that Q_{focus} outperforms A and comes significantly closer to the human learning curve. The differences between the curves for Q_{focus} and A are statistically significant at all points ($\alpha = 0.05$). Apparently the focus heuristic plays a major role in the advantage of A over Q, and by adding that knowledge to generate Q_{focus}, we have generated a system that most closely approximates the human learner of any system we have investigated so far.

Discussion

Our results indicate that relevance knowledge, especially knowledge regarding which sensors are relevant when, is one of the keys to better performance on this domain. Future work will explore adding other elements of A to Q. The rate of human learning on this task is a function of both the amount of strategic knowledge that humans possess on related navigation tasks, as well as the finely honed sensorimotor procedural machinery that makes *effective* use of this information. To make our algorithms competitive with human learning, we are making explicit other forms of navigational knowledge brought to bear by humans and by refining our algorithms for making use of this knowledge. We hope to thereby achieve our goal of developing an algorithm that models human learning on the Navigation task.

Acknowledgements

This research was sponsored by the Office of Naval Research N00014-95-WX30360 and N00014-95-1-0846. We thank Sandy Marshall and William Spears for their helpful comments and suggestions, and especially William Spears for his suggestion to divide the q-learner into two data structures in order to use the focus heuristic. Thanks also 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.* Cambridge, 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.