Comparison of Techniques to Learn Agent Strategies in Adversarial Games*

Shar Whisenhunt and Diane J. Cook
Box 19015
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76006
{shar, cook}@centauri.uta.edu

Abstract

The focus of this project is to develop methodologies for using machine learning techniques in adversarial robot situations. In particular, we are using multiple robots to play a version of the wumpus world game. In this game, one robot represents the agent and a second robot represents the wumpus. Our goal is for the agent robot to make autonomous decisions that allow it to elude the wumpus, grab the gold and win the game.

To achieve this goal, we consult several supervised machine learning algorithms to decide the agent's move. Agent moves are learned from training examples encoding characteristics of the world, the game state, and the predicted wumpus move. In this paper we will compare the performance of a decision tree learner, a naive Bayesian classifier, a backpropagation neural network, and a learning-based belief network on actual wumpus world games.

Introduction

Many adversarial environments exist in which an agent must take the best possible action in order to survive or outwit an opponent. While agent strategies can be learned in such situations, applying learning techniques to real-world applications places constraints on the learner such as being able to deal with uncertainty and learning from few training examples. Uncertainty may arise from a number of sources including incomplete or incorrect information about the world, unexpected changes in the environment, and unknown actions taken by adversaries. The scarcity of training examples may occur when generating training examples is expensive or when a separate training phase is not possible, thus learning must occur in an incremental fashion.

In this project we are interested in using machine learning techniques to automatically select agent moves in a two-player robotic adversarial game. We apply several alternative machine learning methods to to a robotic version of the wumpus world game including C4.5, a naive Bayesian classifier, a backprop neural net, and a learning-based belief network. We measure the performance of these learning systems on simulated games and monitor the performance of each system as the number of available training examples is varied and as the amount of uncertainty in robot sensing changes.

The following section of the paper describes the two-player game we utilize for our experiments, an enactment of the wumpus world game. We then provide a description of the belief network construction and learning algorithms used to suggest agent actions. We evaluate the results of the various approaches using randomly-generated test cases from the wumpus world game.

Wumpus world

The wumpus world is based on an early computer game. The basis for the game is an agent who explores an NxN grid world while avoiding a creature known as the wumpus. Other elements of the world consist of bottomless pits (which do not affect the wumpus), and a bar of gold. The objective of the game is to collect the gold bar, return to the initial grid location [1,1] and exit the cave. The information that the agent senses/receives each turn to aid in locating the gold and avoiding the wumpus and pits, is a five-element percept. If the agent is in a square directly adjacent to a pit it will perceive a breeze. If there is a gold bar in the same location as the agent it will perceive glitter. If the agent is adjacent to the wumpus it will perceive a stench. If the agent runs into a wall it will perceive a bump, and if the agent shoots its arrow and kills the wumpus it will hear a scream. The actions allowed the agent are to move forward, turn right, turn left, grab gold, shoot the arrow, and climb out of the cave. The actions allowed the wumpus are to move forward, turn left, turn right, and do nothing.

Changes to the domain were made to use sonar readings instead of the five element percept, because the robots playing the roles of the agent and the wumpus possess a ring of sonar sensors. The agent is supplied with locations of the pits and the gold bar. Using the sonar grid, the robot can determine where the walls are, can calculate its own position in the world, and can de-

termine whether or not it is in horizontal or vertical alignment with the wumpus.

An obstacle is inserted into the domain to increase the world complexity. If the wumpus and agent are on opposite sides of the obstacle, neither can sense the other robot.

The robots playing the roles of agent and wumpus are Trilobots. The Trilobot robots have been given an eight-element sonar array. The sonar array polls all eight sonar sensors, then rotates 22.5 degrees and polls all sensors again, yielding a total of sixteen readings. If the wumpus is not directly in line (same row or same column of the grid world) as the agent, the agent may not sense the wumpus. A picture of the robots in a portion of the wumpus world can be seen in Figure 1.

The behavioral strategies that the wumpus can employ are to 1) move to a gold bar and circle it clockwise, 2) move to a gold bar and circle it counter-clockwise, 3) attack (move toward) the agent, 4) do nothing, and 5) hide behind an obstacle until the agent comes close and then attack. The wumpus will only follow one of these strategies throughout a given game. We learn a weighted DFA that represents wumpus moves (Peterson & Cook 1998), and predict the next wumpus move at any point during the game using the output of the DFA algorithm.

Learning Agent Moves for Adversarial Games

The goal of this project is to learn agent moves in the two-player wumpus world game. For this learning application we require learning algorithms that can operate in a supervised mode and that can yield good performance in the presence of noise. Although the algorithm will be trained on problems from a simulated game, the learned concepts must perform well in a robotic environment with noisy sensor readings.

We select learning algorithms that meet these criteria and yet represent varied approaches to supervised learning. In particular, we compare the performance of a rule-learner using C4.5 (Quinlan 1993), a statistical approach to learning using a naive Bayesian classifier (Cestnik 1990), a backprop neural network (Rumelhart & McClelland 1986), and Netica, a commercial belief network that learns conditional probability relationships from training examples. The first three learning algorithms are contained in ML2.0, a UTA-created C collection of machine learning routines and performance analysis functions.

We choose to include Netica in our study in particular because we may need to make decisions in our robot domain while some uncertainty exists. We construct a belief network to represent the factors influencing the choice of action for the agent to take. By incorporating the learning algorithm available in the Netica belief network system, the values of the belief net can be learned automatically from randomly-generated training examples. Given uncertainty, we are interested to see if the belief network outperforms the other algorithms which are not set up to handle uncertain conditions. After training, we compare the results of the various algorithms to determine which method most often makes the best decision as to the agent's best move.

We randomly-generate 100 cases to test the first three learning algorithms. Each case is encoded in terms of the world description, the current agent position and wumpus position, and the DFA-predicted wumpus next move. Each case is classified based on the best action that can be taken in the situation. In our modified version of the game, the agent must select an action from the following possibilities: move north, move south, move east, move west, grab the gold, or climb out of the world. Preferred moves bring the agent closer to the gold initially (closer to the [1,1] location if the gold has been grabbed) without moving into the current wumpus square, the pit locations, or the predicted wumpus next location.

When simulated, it is easy to see all the variables that are needed to make a sound decision. We know the position of the all the pits, the obstacle, the gold bar, and the wumpus. However, the robots are not always able to obtain all the information needed to decide the optimal move. For example, the agent may not be able to "see" the wumpus if the wumpus cannot be detected by the agent's sonar sensors. The agent needs to select an action based on this possibly incomplete information.

The structure of our Netica belief net is shown in Figure 2. The gold (x,y) location and agent (x,y) location are provided as input to the network. Based on this information together with the highest-probability predicted wumpus location output by the DFA algorithm, the belief net generates values corresponding to the desirability of each possible action. Using this value and the feasibility of each move (the agent cannot move into pit locations, obstacle locations or current wumpus locations), the final node of the belief network generates a set of values corresponding to the probability of success for each possible agent action. The encoded information described here is available to all of the learning systems except for the causal relationships between parts of the domain.

Netica uses algorithms for fast probabilistic inference in a compiled belief network based on research performed by Spiegelhalter et al (Spiegelhalter *et al.* 1993) and Neapolitan (Neapolitan 1990). Netica's learning algorithm is a type of supervised learning. Each training case represents an example world situation or event. Each feature that can be used to describe the training case becomes a node in the learned network. Although learning of belief networks consists of learning both the structure of the network and the conditional probability relationships at each node, currently Netica performs only parameter learning and assumes the structure is predefined.

Netica assumes that each conditional probability is independent, an assumption that can be fairly well maintained in the wumpus world game. The system will

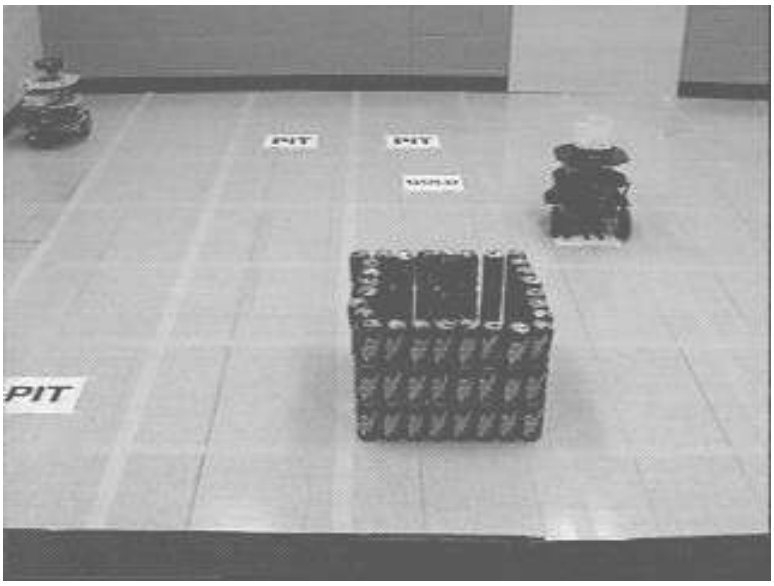

Figure 1: Robotic wumpus world.

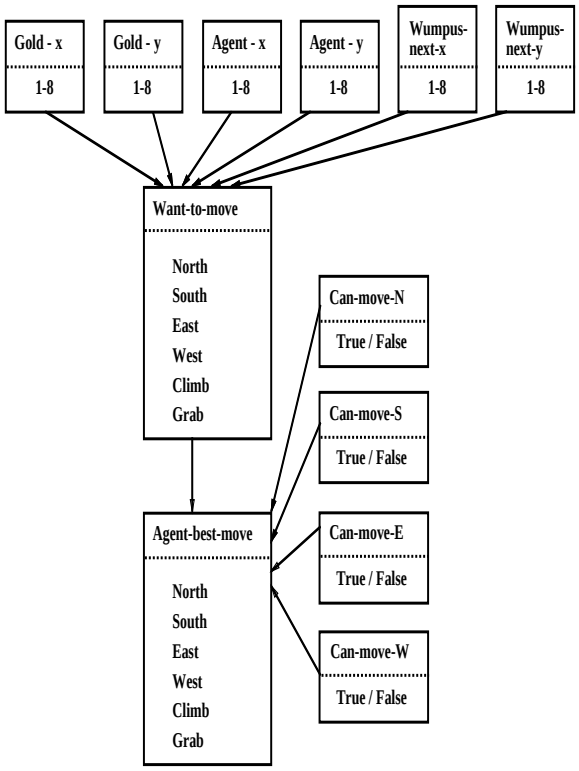

Figure 2: Wumpus world belief network

generate any specified number of training cases, which are classified by the user. Netica will then learn probability relationships that minimize total error based on the sample cases.

Evaluation

To compare the performance of these four systems, we perform ten-fold cross validation on the 100 cases. These training examples include random values for the agent position, the gold position, the wumpus current position, the wumpus's chosen strategy, and six pit locations.

The prediction accuracies for each of the ten trials are shown in Table 1. As can be seen from these results, although the decision tree performs well, the belief network consistently yields the best result and generates the best overall performance. Both C4.5 and Netica performed significantly better than either the backprop algorithm or the Bayesian classifier.

In the first experiment, each system was learning the best moves to reach the gold. To determine how well these systems can learn more complex concepts, we generate 100 training examples, half of which lead the agent to the gold and half of which lead the agent to the exit once the gold is grabbed.

Next, we monitor the algorithms' ability to handle uncertainty. We model the type of uncertainty that will likely occur during robotic execution of the adversarial game. Because of faulty readings, sensor gaps, or obstacle occlusion, the robot may not always accurately sense its own location or the wumpus location. To simulate this uncertainty we inject a "don't know" value for these attributes in a percentage of the training examples. Figure 3 shows that C4.5 and the Bayesian

Trial	C4.5	Bayesian Classifier (BC)	Backprop NN (NN)	Netica (Net)
1	80	60	40	100
2	90	90	20	100
3	90	80	20	90
4	90	70	20	50
5	100	70	40	100
6	80	80	30	100
7	80	70	50	90
8	90	90	60	80
9	90	90	50	100
10	90	100	30	100
Average	**88**	**80**	**36**	**91**
Stat. Sig.	**Bayes – .034854** **Backprop – .000822** **Netica – .302179**	**Backprop – .008055** **Netica – .042075**	**Netica – .003815**	

Table 1: Percentage of cases accurately predicted for each algorithm

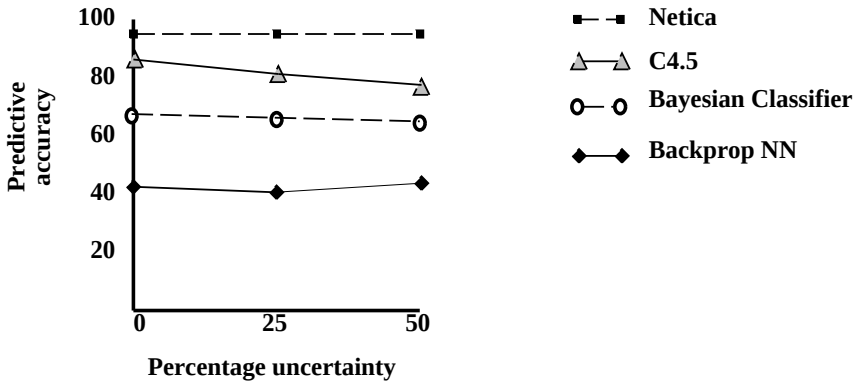

Figure 3: Predictive accuracy with uncertainty

Trial	C4.5	BC	NN	Net
1	80	50	20	100
2	100	90	20	80
3	70	40	60	100
4	70	80	30	90
5	100	50	70	100
6	90	80	30	100
7	90	60	70	100
8	80	80	30	90
9	90	50	50	90
10	90	90	30	90
Average	**86**	**67**	**41**	**94**

Table 2: Percentage of accurated predictions for each algorithm - half gold, half exit

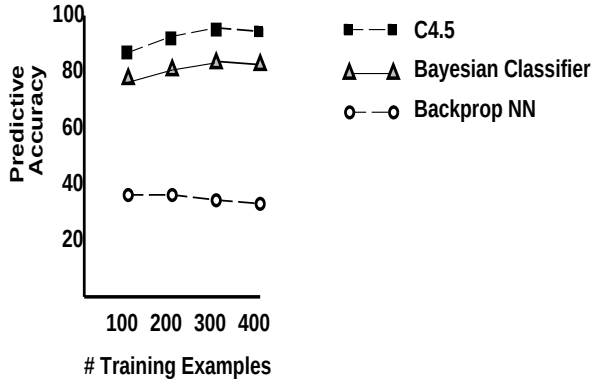

Figure 4: Predictive accuracy as a function of the number of training examples

classifier are adversely affected by an increased amount of uncertainty, while the neural network results jump around and Netica yields steady results even with increased uncertainty. Thus Netica not only yields the best results overall but also demonstrates the greatest stability in the presence of uncertainty.

We are then interested in determining which learning techniques will be affected the most by an increased number of training examples. Because of the training time, we do not include Netica in this experiment. For this experiment, we generate 100, 200, 300, and 400 random training examples using only one goal (grab the gold). As Figure 4 shows, all learning algorithms except the neural network benefit from an increased number of training examples, and the Bayesian classifier realizes the greatest improvement as the number of training examples increases. Because training examples can be expensive to generate in robotic domains, we would select a system such as C4.5 or Netica which can perform well with few training examples but which can also benefit from more training examples when they are available.

Finally, instead of determining which algorithm would perform best for all situations, we attempt to let C4.5 make a decision of which learned output to use for a given game situation. We train the learning system on the 100 game situations (50 gold, 50 exit), using the name of the learning system with the correct answer as the classification for the instance (if multiple algorithms give the right answer, opt for the system with the higher average success rate). When Netica is included as an option, C4.5 always decides to let Netica make the choice. When C4.5 chooses only between itself, the Bayesian classifier, and the neural net, it usually chooses C4.5 and occasionally uses the output of the neural network. Even though the average success of the Bayesian classifier is higher than that of the neural network, its output is never utilized. The average accuracy of the results degrades in this case over the average C4.5 results from 86% accuracy to 83% accuracy, possibly due to the occasional integration of neural network results.

Conclusions

In this paper we presented an application of machine learning techniques to decision making in a robotic adversarial environment. Using pre-classified training examples we are able to learn concepts that yield action decisions conforming closely to the human-selected action in each situation. However, we have also clearly demonstrated that the decision tree and belief network algorithms outperform backprop neural net or Bayesian classifiers on this task. We have also delineated factors about this domain that can affect the selection of a learning algorithm such as sensitivity to number of training examples and ability to handle uncertainty in the data.

There are a number of extensions to this research that we would like to pursue. First, in the same way that a belief network was constructed to pick optimal agent moves, so we would like to construct and learn values for a network that predicts wumpus moves, and compare the predictive accuracy of the belief network with the learned probabilistic DFA. We would also like to train the learning systems on entire game sequences instead of individual moves from the game. Finally, we would like to increase the complexity of the program to allow the wumpus and agent to change strategies during execution of a game.

Acknowledgements

This research is supported by NSF grants HRD-9255016, IRI-9502260 and DMI-9724497.

References

Cestnik, B. 1990. Estimating probabilities: a crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, 174–179.

Neapolitan, R. E. 1990. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms.* John Wiley and Sons.

Peterson, G., and Cook, D. J. 1998. Dfa learning of adversarial strategies. In *to appear in Proceedings of the Florida AI Research Symposium.*

Quinlan, J. R. 1993. *C4.5: Programs For Machine Learning.* Morgan Kaufmann.

Rumelhart, D., and McClelland, J. 1986. *Parallel distributed processing: exploration in the microstructure of cognition, Volumes 1 and 2.* Cambridge, MA: MIT Press.

Spiegelhalter, D. J.; Dawid, A. P.; Lauritzen, S. L.; and Cowell, R. G. 1993. Bayesian analysis in expert systems. *Statistical Science* 8(3):219–283.