Simulation-Assisted Learning by Competition: Effects of Noise Differences Between Training Model and Target Environment †

ALAN C. SCHULTZ (SCHULTZ@AIC.NRL.NAVY.MIL)
CONNIE LOGGIA RAMSEY (RAMSEY@AIC.NRL.NAVY.MIL)
JOHN J. GREFENSTETTE (GREF@AIC.NRL.NAVY.MIL)

Navy Center for Applied Research in Artificial Intelligence (Code 5514),
Naval Research Laboratory, Washington, DC 20375-5000, U.S.A.
(202) 767-2684

Abstract

The problem of learning decision rules for sequential tasks is addressed, focusing on the problem of learning tactical plans from a simple flight simulator where a plane must avoid a missile. The learning method relies on the notion of competition and employs genetic algorithms to search the space of decision policies. Experiments are presented that address issues arising from differences between the simulation model on which learning occurs and the target environment on which the decision rules are ultimately tested. Specifically, either the model or the target environment may contain noise. These experiments examine the effect of learning tactical plans without noise and then testing the plans in a noisy environment, and the effect of learning plans in a noisy simulator and then testing the plans in a noise-free environment. Empirical results show that, while best result are obtained when the training model closely matches the target environment, using a training environment that is more noisy than the target environment is better than using using a training environment that has less noise than the target environment.

1 Introduction

In response to the knowledge acquisition bottleneck associated with the design of expert systems, research in machine learning attempts to automate the knowledge acquisition process and to broaden the base of accessible sources of knowledge. The choice of an appropriate learning technique depends on the nature of the performance task and the form of available knowledge. If the performance task is classification, and a large number of training examples are available, then inductive learning techniques (Michalski, 1983) can be used to learn classification rules. If there exists an extensive domain theory and a source of expert behavior, then explanation-based methods may be applied (Mitchell et. al, 1985). For many interesting sequential decision tasks, there exists neither a database of examples nor a reliable domain theory. In these cases, one method for manually developing a set of decision rules is to test a hypothetical set of rules against a simulation model of the task environment, and to incrementally modify the decision rules on the basis of the simulated experience. This paper presents some initial efforts toward using machine learning to automate the process of learning sequential tasks with a simulation model.

Sequential decision tasks may be characterized by the following general scenario: A decision making agent interacts with a discrete-time dynamical system in an iterative fashion. At the beginning of each time step, the agent observes a representation of the current state and selects one of a finite set of actions, based on the agent's decision rules. As a result, the dynamical system enters a new state and returns a (perhaps null) payoff. This cycle repeats indefinitely. The objective is to find a set of decision rules that maximizes the expected total payoff.[1] Several sequential decision tasks have been investigated in the machine learning literature, including pole balancing (Selfridge, Sutton & Barto, 1985), gas pipeline control (Goldberg, 1983), and the animat problem (Wilson, 1987). For many interesting problems, including the one considered here, payoff is delayed in the sense that non-null payoff occurs only at the end of an episode that may span several decision steps. In fact, the paradigm is quite broad since it includes any problem solving task by defining the payoff to be positive for any goal state and null for non-goal states (Barto, Sutton & Watkins, 1989).

† To appear in the Proceedings of the Seventh International Conference on Machine Learning, June 1990.

[1] If payoff is accumulated over an infinite period, the total payoff is usually defined to be a (finite) time-weighted sum (Barto et. al, 1989).

The experiments described here reflect two important methodological assumptions:

1. Our learning system is designed to continue learning indefinitely.
2. Since learning may require experimenting with decision rules that might occasionally produce unacceptable results if applied to the real world, we assume that hypothetical rules will be evaluated in a simulation model.

In this methodology, a set of rules is periodically extracted from the learning system to represent the learning system's current plan. This plan is tested in the target environment, and the resulting performance is plotted in a learning curve. Making a clear distinction between the simulation model used for training and the target environment used for testing suggests a number of experiments that measure the effects of differences between the training model and the target environment. Simulation models have played an important role in earlier machine learning efforts (Goldberg, 1983; Booker, 1982, 1988; Wilson, 1985; Buchanan et al., 1988); however, these works do not address the issue of validation of the simulation model with respect to a real target system. The experiment described here represents a step in this direction. Specifically, differences between the training model (the simulator) and the target environment with respect to noise are examined.

2 The Evasive Maneuvers Problem

The experiments described here concern a particular sequential decision task called the *Evasive Maneuvers (EM)* problem, inspired in part by Erickson and Zytkow (1988). The tactical objective is to maneuver a plane to avoid being hit by an approaching missile. The missile tracks the motion of the plane and steers toward the plane's anticipated position. The initial speed of the missile is greater than that of the plane, but the missile loses speed as it maneuvers. If the missile speed drops below some threshold, it loses maneuverability and drops out of the sky. It is assumed that the plane is more maneuverable than the missile. There are six sensors that provide information about the current tactical state:

1. *last-turn*, the current turning rate of the plane;
2. *time*, a clock that indicates time since detection of the missile;
3. *range*, the missile's current distance from the plane;
4. *bearing*, the direction from the plane to the missile;
5. *heading*, the missile's direction relative to the plane; and
6. *speed*, the missile's current speed measured relative to the ground.

Although other sensors could be used, these sensors represent a minimal, realistic set that might be available from the pilots point of view.

Finally, there is a discrete set of actions available to control the plane. In this study, we consider only actions that specify discrete turning rates for the plane.[2] The learning objective is to develop a *tactical plan*, i.e., a set of decision rules that map current sensor readings into actions that successfully evade the missile whenever possible.

The EM problem is divided into *episodes* that begin when the threatening missile is detected and that end when either the plane is hit or the missile is exhausted.[3] It is assumed that the only feedback provided is a numeric payoff, supplied at the end of each episode, that reflects the quality of the episode with respect to the goal of evading the missile. Maximum payoff is given for successfully evading the missile, and a smaller payoff, based on how long the plane survived, is given for unsuccessful episodes.

The EM problem is clearly a laboratory-scale model of realistic tactical problems. Nevertheless, it includes several features that make it a challenging machine learning problem:

- a weak domain knowledge (e.g., no predictive model of missile);
- incomplete state information provided by discrete (possibly, noisy) sensors;
- a large state space; and, of course,
- delayed payoff.

The following sections present one approach to addressing these challenges.

3 SAMUEL on EM

SAMUEL[4] is a system designed to explore competition-based learning for sequential decision tasks (Grefenstette, 1989). SAMUEL consists of three major components: a problem specific module, a performance module, and a learning module. The problem specific module consists of the task environment simulation, or world model (in this case, the EM model), and its interfaces. The performance module is called CPS (Competitive Production System), a production system that interacts with the world model by reading sensors, setting control variables, and obtaining payoff from a critic. In addition to matching, CPS implements conflict resolution as a competition among rules based on rule strength and performs credit assignment based on payoff (Grefenstette, 1988). The learning module uses a genetic

[2] The current statement of the problem assumes a two-dimensional world. Future experiments will adopt a three-dimensional model and will address problems with multiple control variables, such as controlling both the direction and the speed of the plane.

[3] For the experiments described here, the missile began each episode at a fixed distance from the plane, traveling toward the plane at a fixed speed. The direction from which the missile approached was selected at random.

[4] SAMUEL stands for Strategy Acquisition Method Using Empirical Learning. The name also honors Art Samuel, one of the pioneers in machine learning.

algorithm to develop tactical plans, expressed as a set of condition-action rules. Each plan is evaluated on a number of tasks in the world model. As a result of these evaluations, genetic operators, such as *crossover* and *mutation*, produce plausible new plans from high performance parents. More detailed descriptions of SAMUEL appear in (Grefenstette, 1989; Grefenstette, Ramsey & Schultz, 1990).

The design of SAMUEL owes much to Smith's LS-1 system (Smith, 1980), and draws on some ideas from classifier systems (Holland, 1986). In a departure from these earlier genetic learning systems, SAMUEL learns plans consisting of rules expressed in a high level rule language.

The rule language allows four types of sensors:

1. *linear*, where the condition specifies an upper and lower bound for the linear ordered values;
2. *cyclic*, which are like linear, except that the values wrap around from the upper bound to the lower bound;
3. *structures*, where the sensor specifies a list of values, and the condition matches if the sensor's current value occurs in a subtree labeled by one of the values in the list; and
4. *pattern*, where the sensor specifies a pattern over the alphabet {0, 1, #}, as in classifier systems.

In the EM domain, only linear and cyclic type sensors are used. An example of a rule for EM follows:

```
if  (and (last-turn 0  45) (time 4  14) (range 500  1400)
         (heading 330  90) (speed 50  850))
then (and (turn 90))
strength 750
```

Each condition (*if* part) specifies a range over the named sensor, and each action (*then* part) specifies the value for the named control variable. The strength is an estimate of the rule's utility and is used for conflict resolution (Grefenstette, 1988).

The use of a high level language for rules offers several advantages over low level binary pattern languages typically adopted in genetic learning systems (Smith, 1980; Goldberg, 1983). First, it makes it easier to incorporate existing knowledge, whether acquired from experts or by symbolic learning programs. Second, it is easier to transfer the knowledge learned to human operators. Third, it makes it possible to combine empirical methods such as genetic algorithms with analytic learning methods that explain the success of the empirically derived rules (Gordon & Grefenstette, 1990).

4 Evaluation of the Method

This section presents an empirical study of the performance of SAMUEL on the EM problem with respect to the differences between the simulation model in which the knowledge is learned and the target environment in which the learned knowledge will be used.

4.1 Experimental Design

The learning curves shown in this section reflect our assumptions about the methodology of simulation-assisted learning. In particular, we make a distinction between the world model used for learning and the target environment. Let E denote the target environment for which we want to learn decision rules. Let M denote a simulation model of E that can be used for learning. The assumption is that learning continues indefinitely in the background using system M, while the plan being used on E is periodically updated with the current hypothetical plan of the learning system. The genetic algorithm in SAMUEL evaluates the fitness of each plan in its population by measuring its performance on a number of episodes (currently, 10 episodes per plan) in the training model M. A plan's fitness determines its reproductive probability for the next generation. At periodic intervals (currently, 10 generations), a single plan is extracted from the current population to represent the learning system's current hypothetical plan. The extraction is accomplished by re-evaluating the top 20% of the current population on 100 randomly chosen episodes on the simulation model M. The plan with the best performance in this phase is designated the current hypothesis of the learning system. This plan is tested in the environment E for 100 randomly chosen problem episodes. The plots show the sequence of results of testing on E, using the current plans periodically extracted from the learning system. Distinguishing these two systems permits the study of how well the learned plans behave if E varies significantly from M, as is likely in practice.

Because SAMUEL employs probabilistic learning methods, all graphs represent the mean performance over 20 independent runs of the system, each run using a different seed for the random number generator. When two learning curves are plotted on the same graph, a vertical line between the curves indicates that there is a statistically significant difference between the means represented by the respective plots (with significance level $\alpha = 0.05$) at that point on the curves. This device allows the reader to see significant differences between two approaches at various points during the learning process. We feel that this is a better way to compare learning curves for continuously learning systems than, say, running the two systems for a fixed amount of time and comparing the performance of the final plans.

4.2 Sensitivity to Sensor Noise

Noise is an important topic in machine learning research, since real environments can not be expected to behave as nicely as laboratory ones. While there are many aspects of a real environment that are likely to be noisy, we can identify three major sources of noise in the kinds of sequential decision tasks for which SAMUEL is designed: sensor noise, effector noise, and payoff noise. Sensor noise refers to errors in sensor data caused by imperfect sensing devices. For example, if the radar indicates that the range to an object is 1000 meters when

in fact the range is 875 meters, the performance system has received noisy data. Effector noise refers to errors arising when effectors fail to perform the action indicated by the current control settings. For example, in the EM world, an effector command might be (turn 45), meaning that the effectors should initiate a 45 degree left turn. If the plane turns at a different rate, the effector has introduced some noise. An additional source of noise can arise during the learning phase, if the critic gives noisy feedback. For example, a noisy critic might issue a high payoff value for an episode in which the plane is hit.

While all of these types of noise are interesting, we restrict our attention to the noise caused by sensors. To test the effects of sensor noise on SAMUEL, two environments were defined. In one environment, the sensors are noise-free. In the second environment, noise is added to each of the four external sensors that indicate the missile's range, bearing, heading, and speed. Noise consists of a random draw from a normal distribution with mean 0.0 and standard deviation equal to 10% of the legal range for the corresponding sensor. The resulting value is then discretized according to the defined granularity of the sensor. For example, suppose the missile's true heading is 66 degrees. The noise consists of a random draw from a normal distribution with standard deviation 36 (10% of 360 degrees), resulting in a value of, say, 22. The noisy result, 88, is then discretized to the nearest 10 degree boundary (as specified by the granularity of the heading sensor), and the final sensor reading is 90. As this example shows, the amount of noise in this environment is rather substantial.

Given the noisy environment and the noise-free environment, there are four possible experimental conditions, depending on which environment is used for the simulation model (M) and which is used for the target environment (E). For each experimental condition, the genetic algorithm was executed for 200 generations, and the results are shown in Figures 1 and 2.

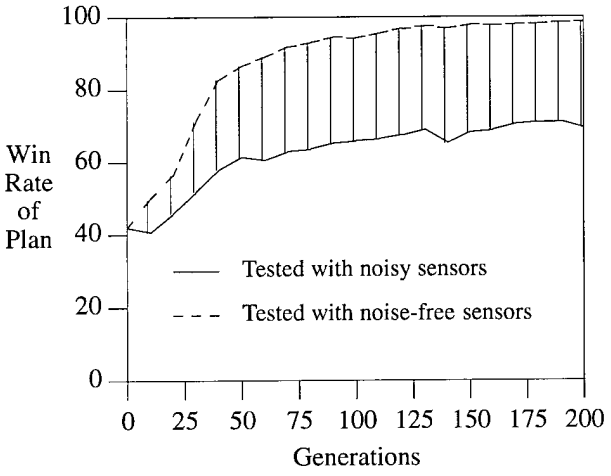

Figure 1: Learning with Noise-Free Sensors

In Figure 1, training was performed in the model with noise-free sensors, and the resulting plans were tested in the environment with noise-free sensors (dashed curve) and noisy sensors (solid curve). In Figure 2, training was performed in the model with noisy sensors, and again tested with both noise-free sensors (dashed curve) and noisy sensors (solid curve).

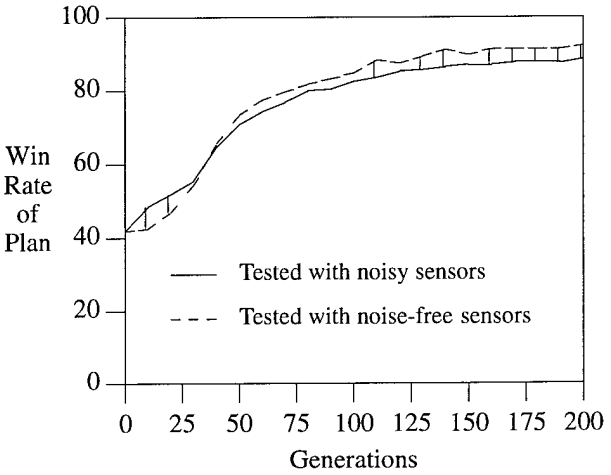

Figure 2: Learning with Noisy Sensors

Figure 1 shows that plans learned with noise-free sensors perform significantly worse throughout the learning period when the testing environment contains noise (solid curve) than when the testing environment is noise-free (dashed curve). For example, after 200 generations, the current tactical plan evades the missile about 98% of the time when the sensors are noise-free, but only about 70% of the time when the sensors are noisy. On the other hand, Figure 2 shows that the plans learned under noisy conditions perform fairly well in both target environments. After 200 generations, the current tactical plan evades the missile about 88% of the time when the sensors are noisy, and about 92% when the sensors are noise-free. Clearly, more robust rules are being learned, at a cost of slower improvement.

By comparing the two dashed curves (or the two solid curves) in Figures 1 and 2, it may be concluded that, for a fixed target environment, SAMUEL learns best when the training environment matches the target environment. However, this ideal case will not generally be realized in practice, especially if the target is a real-world system and the training model is a simulation. Our results show that using a training environment that is less regular (in this case, more noisy) than the target environment is better than having a training model with spurious regularities (e.g., noise-free sensors) that do not occur in the target environment.

5 Summary and Further Research

One important lesson of the empirical study is that SAMUEL is an opportunistic learner, and will tailor the plans that it learns to the regularities it finds in the training model. It follows that the closer the training model matches the conditions expected in the target environment -- in terms of sensor noise -- the better the learned plans will be. In the absence of a perfect match between training model and target environment, it is better to have too little regularity in the training model than too much.

This study illustrates just one of many investigations one might pursue in assessing the effects of differences between the model and the target environment. In another study, we have examined the effect of differences in the initial conditions (i.e., the missile's initial range, speed, and heading) between the training model and the target environment. Preliminary results support the general conclusion reported here -- that it is far less risky to have a training model with overly general initial conditions than to have one with overly restricted initial conditions (Grefenstette et. al, 1990).

Current efforts are also aimed at augmenting the task environment to test SAMUEL's ability to learn tactical plans for more realistic scenarios. Multiple incoming threats will be considered, as well as multiple control variables (e.g., accelerations, directions, weapons, etc.).

As simulation technology improves, it will become possible to provide learning systems with high fidelity simulations of tasks whose complexity or uncertainty precludes the use of traditional knowledge engineering methods. No matter what the degree of sophistication of the simulator, it will be important to assess the effects on any learning method of the differences between the simulation model and the target environment. These initial studies with a simple tactical problem have shown that it is possible for learning systems based on genetic algorithms to effectively search a space of knowledge structures and discover sets of rules that provide high performance in a variety of target environments. Further developments along these lines can be expected to reduce the manual knowledge acquisition effort required to build systems with expert performance on complex sequential decision tasks.

5.0.1 References

Barto, A. G., R. S. Sutton and C. J. C. H. Watkins (1989). Learning and sequential decision making. COINS Technical Report, University of Massachusetts, Amherst.

Booker, L. B. (1982). *Intelligent behavior as as adaptation to the task environment.* Doctoral dissertation, Department of Computer and Communications Sciences, University of Michigan, Ann Arbor.

Buchanan, B. G. J. Sullivan, T. P. Cheng and S. H. Clearwater (1988). Simulation-assisted inductive learning. *Proceedings Seventh National Conference on Artificial Intelligence.* (pp. 552-557).

Erickson, M. D. & J. M. Zytkow (1988). Utilizing experience for improving the tactical manager. *Proceedings of the Fifth International Conference on Machine Learning.* Ann Arbor, MI. (pp. 444-450).

Goldberg, D. E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and machine learning,* Doctoral dissertation, Department Civil Engineering, University of Michigan, Ann Arbor.

Gordon, D. G & J. J. Grefenstette (1990). Explanations of empirically derived reactive plans. *Proceedings of the Seventh International Conference of Machine Learning,* Austin, Texas.

Grefenstette, J. J. (1988). Credit assignment in rule discovery system based on genetic algorithms. *Machine Learning, 3(2/3),* (pp. 225-245).

Grefenstette, J. J. (1989). Incremental learning of control strategies with genetic algorithms *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann. (pp. 340-344).

Grefenstette, J. J., Connie Loggia Ramsey, and Alan C. Schultz (1990). Learning sequential decision rules using simulation models and competition. To appear in *Machine Learning.*

Holland J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R.S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach,* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Michalski, R. S. (1983). A theory and methodology for inductive learning. *Artificial Intelligence, 20(2),* (pp. 111-161).

Mitchell, T. M., S. Mahadevan and L. Steinberg (1985). LEAP: A learning apprentice for VLSI design. *Proc. Ninth IJCAI,* (pp. 573-580). Los Angeles: Morgan Kaufmann.

Selfridge, O., R. S. Sutton and A. G. Barto (1985). Training and tracking in robotics. *Proceedings of the Ninth International Conference on Artificial Intelligence.* Los Angeles, CA. August, 1985.

Smith, S. F. (1980). *A learning system based on genetic adaptive algorithms,* Doctoral dissertation, Department of Computer Science, University of Pittsburgh.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. *Proceedings of the International Conference Genetic Algorithms and Their Applications* (pp. 16-23). Pittsburgh, PA.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning, 2(3),* (pp. 199-228).