Do We Really Need to Estimate Rule Utilities in Classifier Systems?

Lashon B. Booker
The MITRE Corporation
1820 Dolley Madison Blvd
McLean, VA 22102-3481

Abstract

Classifier systems have traditionally used explicit measures of utility (strength, predicted payoff, accuracy, etc.) to quantify the performance and fitness of classifier rules. Much of the effort in designing and implementing these systems has focused on getting these utilities "right". One alternative worth exploring is the idea of using endogenous fitness; that is, reinforcing successful performance with "resources" that rules need in order to reproduce. Under this regime, the best rules are those that accumulate the most resources over their lifetime and, consequently, have the most offspring. This paper describes a classifier system designed along these lines. Rules have no associated utility measure, just a resource reservoir. When enough resources have been accumulated, the rule reproduces and the reservoir is reduced. Preliminary tests of this system on the multiplexor problem show that it performs as well as utility-based classifier systems such as XCS.

1 INTRODUCTION

One of the most attractive aspects of the classifier system framework is the way it treats rules as tentative hypotheses subject to testing and confirmation via experience with the environment (Holland, 1986). A key requirement underlying this capability is a computational approach to assessing the strength of each hypothesis. In classifier systems, the strength of each hypothesis is determined from experienced-based predictions of rule performance. Additionally, hypothesis strength provides a sound basis for assessing the fitness of rules, making rules a source of building blocks for generating plausible new hypotheses. Classifier system implementations have traditionally put this idea into practice by calculating parameters — such as predicted payoff, accuracy, payoff-derived strength, etc. — that explicitly estimate the utility and fitness of rules. We are investigating a classifier system that determines the utility and fitness of rules endogenously, without computing explicit estimates.

This research was prompted in part by difficulties encountered over the years trying to devise quantitative performance measures and fitness functions for traditional classifier systems. In a complex and uncertain environment, a single measure of what makes a hypothesis useful or reliable can be hard to come by. Sometimes the system might need classification hypotheses that minimize the total number of errors, or perhaps just hypotheses that eliminate false positive errors. In other cases the challenge could be a set covering problem requiring a robust covering of the inputs subject to various constraints. Designing a single performance measure that accommodates all of the relevant performance requirements, including their context dependent interactions, can be a significant engineering challenge.

Holland recognized that many of the adaptive interactions that arise in classifier systems can be studied more directly in a simpler framework. The Echo class of models (Holland, 1994) provides such a setting. Echo offers a framework in which simply-defined agents interact in carefully structured ways that are more concrete than the general-purpose, pattern-directed procedures used in classifier systems. One of the interesting differences between Echo and classifier systems is that there are no explicit fitness functions in Echo. The fitness of agents in Echo is determined endogenously. Agents interact with each other and with the environment to acquire the resources needed to survive and reproduce. In this way, fitness is a function of the overall context the agent finds itself in, but that context never has to be distilled into quantitative decision criteria.

We hypothesize that endogenous fitness can provide a similar advantage in the classifier system framework. This paper introduces ideas about how to implement endogenous fitness in classifier systems, and gives preliminary results on how well those ideas work. We begin with a brief discussion of why we expect endogenous fitness to be useful in classifier systems. That discussion is followed by a description of our initial implementation, and results from experiments with the Boolean multiplexor problem.

2 ENDOGENOUS FITNESS IN CLASSIFIER SYSTEMS

Endogenous fitness is not a magic remedy that automatically will generate classifier systems having whatever properties we desire. However, it is a useful tool that provides new ways to manage some tricky implementation issues. Our focus in this paper is on the issue of generalization accuracy. As Wilson (1995) has pointed out, traditional classifier systems have no way of distinguishing between an accurate classifier having some modest payoff value, and an inaccurate classifier whose average payoff is equal to that value. Moreover, the traditional classifier systems have no explicit mechanisms in place to promote the discovery and proliferation of accurate rules. As a result, overly general rules continually threaten to take over the population, since they match more messages than their less frequently activated, but more accurate, counterparts.

Though measuring the accuracy of a rule can itself be a tricky endeavor (Horn *et al*, 1994), there are ways to define rule utilities so that overly general rules are less fit and less likely to dominate the population. For example, one of Holland's earliest discussions of classifier systems emphasized the importance of using prediction errors to identify rules that over estimate their expected payoff (Holland, 1976). In addition, Booker (1989) defined a consistency parameter, based on the mean-squared prediction error, that was used to attenuate rule fitness and provide an advantage for accurate rules. Most recently, Wilson (1995) has devised the XCS system that uses a fitness criterion based solely on minimizing prediction error. The success of XCS demonstrates how important it is to address the accuracy issue in any successful classifier system implementation.

The rationale for using endogenous fitness to address the problem of generalization accuracy is based on the following intuition: If rules need resources to reproduce, and accurate rules accumulate resources at a faster rate than inaccurate rules, then accurate rules will tend to take over the population. The key is to find simple ways to adjust the relative rate at which accurate and inaccurate rules accumulate resources. Our approach therefore focuses on how to design such a competition for resources between accurate and inaccurate rules.

To begin with a simple case, assume that we are dealing with a stimulus-response problem in which the rules advocating an action on a given time step (i.e., the action set) take all of the credit for whatever reinforcement (positive or negative) is received. Assume further that each reinforcement event leads to the distribution of some nominal resources among those rules, and that the acquired resources accumulate over time in internal reservoirs. Following the way resources are used in Echo (Holland, 1994), we specify that a rule needs to accumulate resources in excess of some threshold level before it is allowed to reproduce. Note that the expected net accumulation of resources, reflecting the net balance of positive and negative reinforcement received, does not depend on how frequently a rule is active. It depends most directly on the probability of reinforcement and, therefore, on the probability that the rule is "correct". Consequently, there is some hope that we can sidestep the issue faced by systems employing explicit fitness functions which must try to compensate for the fact that general rules typically have more opportunities to reproduce than specialized rules.

While this intuitive picture of how to proceed is fairly straightforward, there are many pragmatic details that need to be sorted out. The most important details are related to the parameters governing the rate of accumulation of resources and the frequency of reproduction. Mathematically speaking, this approach tries to exploit the stochastic properties of the increases and decreases in reservoir levels. The sequences of changes in reservoir levels can be viewed as a random walk. Many of the analytical results concerning random walks are counterintuitive, suggesting that analysis is ultimately a better guide than intuition in trying to understand these problems. As a prelude to a formal analysis, we have conducted a series of empirical experiments to help determine an appropriate way to pose the problem in the classifier system setting. The remainder of this paper describes those experiments and the classifier system implementations that appear to be on the right track.

3 SYSTEM DESCRIPTION

The classifier system used in this study has a fairly conventional design, borrowing elements from previous work on GOFER (Booker, 1988; Booker, 1989) and XCS (Wilson, 1995). The population has a fixed size N. Each classifier in the population has a single condition on the left side and a single action on the right side. In a departure from previous systems, there are no performance parameters associated with classifiers. Instead, each classifier ξ has two associated reservoirs: the $\Gamma_+(\xi)$ reservoir that stores resources obtained from positive reinforcement, and the $\Gamma_-(\xi)$ reservoir that stores resources obtained from negative reinforcement. In our experiments the reservoirs were initialized to be empty, and the initial classifiers were generated at random. The only other parameter stored with each classifier is its age, which is used in the procedure for deleting classifiers.

3.1 PERFORMANCE SYSTEM

The system performance cycle is fairly routine. For each input message i, the system first determines the set of classifiers $\mathbf{M}$ eligible to classify the message. Matching classifiers are always included in $\mathbf{M}$. Following the procedures in GOFER, if there are fewer than N_c matching classifiers available, classifiers with the highest partial match scores are deterministically selected to fill out $\mathbf{M}$. We use the simple partial match score

$$\mu(\xi, i) = \begin{cases} s + l & \text{if } \xi \text{ matches the message } i \\ l - n & \text{otherwise} \end{cases}$$

where l is the length of the input condition in ξ, s is the specificity[1] , and n is the number of positions where the condition doesn't match the message.

For each action a represented in $\mathbf{M}$, the system computes an *action mandate* that is similar in intent to the system prediction computed in XCS. The action mandate is supposed to capture the system's knowledge about the likelihood of receiving positive reinforcement if action a is chosen. There are several ways this could be computed. We currently use the relative amount of resources in the $\Gamma_+(\xi)$ and $\Gamma_-(\xi)$ reservoirs as an indicator of expected reinforcement. Each classifier ξ in $\mathbf{M}$ contributes

$$\lambda(\xi) = \frac{|\Gamma_+(\xi)|}{(|\Gamma_+(\xi)| + |\Gamma_-(\xi)|)}$$

to the mandate for its action. The rationale for this particular approach is to give more weight to those actions that, based on previous experience, have the highest likelihood of being followed by positive reinforcement. This contribution from each rule is added to an *action selection array* and, as in XCS, an action is selected using one of many possible selection methods. The members of $\mathbf{M}$ that agree with the selected action constitute the *action set* $\mathbf{A}$. The system then sends that action to the effectors, and the environment may respond with reinforcement.

[1]The number of non-#'s in the input condition

3.2 REINFORCEMENT

For convenience, we assume the external reinforcement values have been scaled so that positive reinforcement is indicated by positive numbers, negative reinforcement is indicated by negative numbers, and zero indicates no reinforcement. So far we have used the same magnitude for all positive and negative reinforcement signals, though signals with different magnitudes could presumably be used to code for more graded, relative reinforcement levels if desired. When reinforcement is received, a fixed amount of resource $\mathbf{R} \geq 0$ is made available to the classifiers in the action set. As in the GOFER system, each action set classifier is eligible to receive a share of the available resources. We have experimented with several ways of allocating the resource among the classifiers in $\mathbf{A}$. The most effective method so far divides the resource based on how strongly each classifier indicates the likelihood of positive or negative reinforcement. This likelihood is estimated by

$$\lambda_*(\xi) = \frac{|\Gamma_*(\xi)|}{(|\Gamma_+(\xi)| + |\Gamma_-(\xi)|)}$$

where $\Gamma_*(\xi)$ is the reservoir containing the largest amount of resources. Each classifier in $\mathbf{A}$ receives a share of the resource given by

$$\rho(\xi) = \frac{\lambda_*(\xi)\mathbf{R}}{\Lambda}$$

where $\Lambda = \sum_{\xi \in \mathbf{A}} \lambda_*(\xi)$ is the total likelihood in $\mathbf{A}$. When the reinforcement is positive, $\rho(\xi)$ is added to $\Gamma_+(\xi)$; otherwise, it is added to $\Gamma_-(\xi)$.

3.3 RULE DISCOVERY

After the rule reservoirs have been updated, any classifier in the match set having a sufficient excess of resources in the $\Gamma_*(\xi)$ reservoir becomes eligible to reproduce. An excess of resources is indicated by

$$|\Gamma_+(\xi) - \Gamma_-(\xi)| > \tau$$

for some threshold τ. If there is more than one eligible classifier, all eligible classifiers are designated as parents and allowed to produce one copy of themselves.

Parents then have their $\Gamma_*(\xi)$ reservoir decremented by τ, which can be viewed as the cost of generating an offspring. The reproduced copies are modified by mutation and crossover, and the resulting offspring are inserted into the population. Classifiers are stochastically selected for deletion based on their age, so that older classifiers are more likely to be chosen. Note that this scheme triggers rule discovery without using any extra bookkeeping or monitoring of the state of the system.

4 EMPIRICAL RESULTS

As a simple initial test of how well this system works, we apply it to the familiar Boolean multiplexor problem. The value of the multiplexor function is computed by using the first n bits of an l-bit string as an index. That index selects one of the remaining 2^n bits in the string (where $n+2^n=l$), and the selected bit is returned as the function value. We restrict our attention to the case $n=2$ (the 6-bit multiplexor). For this problem, the following set of rules provides a complete solution:

```
000### ==> 0   01#0## ==> 0
001### ==> 1   01#1## ==> 1
10##0# ==> 0   11###0 ==> 0
10##1# ==> 1   11###1 ==> 1
```

Each of these rules is maximally general in the sense that no more #'s can be added to their input conditions without introducing performance errors.

In all experiments, unless otherwise noted, the system constants were as follows: $N = 400$, $N_c = 16$, $\mathbf{R} = \pm 500$, $\tau = 1000$, initial reservoir levels of 0, a mutation rate of 0.01, and a crossover rate of 1.0. Performance was measured by the proportion of correct decisions over a learning epoch consisting of 50 randomly generated input strings. Each experiment was run for 200 epochs (10,000 input strings). In order to facilitate comparisons with XCS on this problem, we used Wilson's (1995) action-selection regime. This regime alternates probabilistically between "exploit" mode — where the system chooses the best action as indicated by the action mandate array — and "explore" mode where the system chooses an action at random[2]. No reservoir adjustments or rule discovery operations occur in exploit mode. All results are averaged over 10 independent runs.

[2]During explore trials we record a correct decision whenever the system would have made the correct exploit decision

4.1 TEST OF THE BASIC SYSTEM

The results of applying the system described above to the 6-bit multiplexor problem are shown in Figure 1.

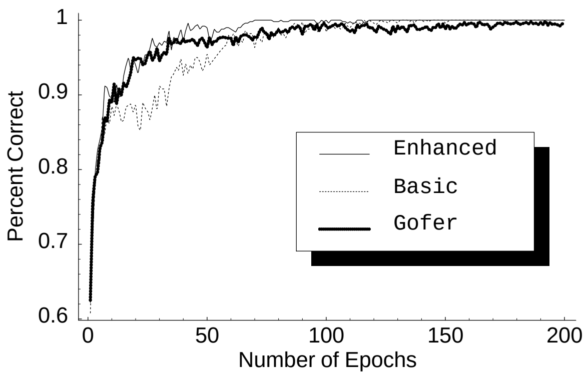

Figure 1: Performance on 6-bit Multiplexor

This basic configuration performs slightly worse than the more conventional GOFER approach over the first 50 epochs, but quickly achieves more favorable performance. At the end of 200 epochs, performance was steady at 100% correct (compared to 99.6% for GOFER). Direct comparison with the GOFER performance curve is somewhat misleading, since GOFER used a roulette-wheel action selection scheme that allowed learning on every trial. Moreover, GOFER only tried to learn the 8 maximally general classifiers that are always correct. Here, in order to assess generalization accuracy, we make our system learn all 16 maximally general classifiers, both those that are 100% correct and those that are 100% incorrect. Nevertheless, the performance of the endogenous fitness approach keeps pace with GOFER on this task and is similar to results reported for XCS.

Figure 2 shows a sample from one of the population of classifiers at the end of the experiment. The numbers in parentheses after each rule give the number of rule instances and the average reinforcement received when that rule controls the system response[3]. The system typically learns many of the maximally general classifiers, but the rules cover the problem landscape in an inefficient and somewhat unpredictable manner. In this regard, performance is less desirable than the results that have been reported for XCS.

[3]The statistics about reinforcement were computed for diagnostic purposes only. They were not used by the performance system.

```
(State = #01#1#) ==> 0 (12,-1000.0)
(State = #01#1#) ==> 1 (12,1000.0)
(State = #1#1#1) ==> 0 (8,-1000.0)
(State = #1#1#1) ==> 1 (16,1000.0)
(State = 000###) ==> 1 (24,-1000.0)
(State = 000#0#) ==> 0 (8,1000.0)
(State = 001###) ==> 1 (10,1000.0)
(State = 01#0##) ==> 0 (13,1000.0)
(State = 01#0##) ==> 1 (18,-1000.0)
(State = 01#1##) ==> 0 (29,-1000.0)
(State = 01#1##) ==> 1 (16,1000.0)
(State = 1###00) ==> 0 (12,1000.0)
(State = 1###00) ==> 1 (14,-1000.0)
(State = 1###11) ==> 0 (13,-1000.0)
(State = 1###11) ==> 1 (9,1000.0)
(State = 11###0) ==> 0 (14,1000.0)
(State = 11###0) ==> 1 (12,-1000.0)
```

Figure 2: Rules learned in the basic endogenous fitness scheme

4.2 AN ENHANCED VARIATION

One of the reasons that the basic system fails to consistently find all of the maximal generalizations is that the selection pressure against overly general classifiers and inefficient covers is weaker here than in XCS. We can improve the performance of the system by finding a way to provide more direct guidance about how to generate a good cover.

Smith *et al* (1993) describe an intriguing approach to generating selection pressures for efficient covers. Their technique is based on a simple model of the immune system in which antibodies compete with each other to match antigens. Both antigens and antibodies are represented as binary strings. The fitness of each antibody is determined by the number of times it is selected as the winner of a competition. Experiments show that, under this fitness regime, the GA generates and maintains a diverse population of antibodies that covers the set of antigens. Moreover, each antibody has a frequency in the population proportional to the frequency with which the corresponding antigens it covers are sampled. An analysis of this mechanism shows that it is strongly related to the familiar fitness-sharing schemes often used in GA implementations to solve multimodal optimization problems.

We can exploit this approach for our purposes by using the probability of winning such a competition to help bias the distribution of resources in $\mathbf{A}$. More specifically, assume that we want to model the results of the following competition for each input message:

1. Select a random sample of size σ without replacement from the population of classifiers.
2. Choose the classifier $\xi \in \mathbf{M}$ with the largest value of $\lambda_*(\xi)$ in the sample as the winner. Ties are broken at random. If there are no classifiers from $\mathbf{M}$ in the sample, then there is no winner and we repeat step 1.
3. Add $\lambda_*(\xi)$ to the total competition score for ξ.
4. Repeat the process for several cycles.

Smith *et al* show that the probability of winning this kind of competition can be expressed in terms of hypergeometric probabilities. The hypergeometric distribution $h(k, r, n, m)$, given by

$$h(k, r, n, m) = \frac{\binom{m}{k}\binom{n-m}{r-k}}{\binom{n}{r}}$$

is used to compute the probability that exactly k red balls will be in a sample of size r, drawn without replacement from a population of size n that contains m red balls. In order for a classifier ξ to win the competition for a given message, no classifier ξ' with $\lambda_*(\xi') > \lambda_*(\xi)$ can occur in the sample. Moreover, whenever there are classifiers ξ' with $\lambda_*(\xi') = \lambda_*(\xi)$ in the sample, ξ has to be the one chosen by the tie-breaking procedure. The probability that these events all occur is given by

$$\hbar(\xi) = \frac{h(0, r, N, G(\xi))(1 - h(0, r, N - G(\xi), F(\xi)))}{F(\xi)}$$

where $F(\xi)$ is the number of classifiers ξ' in $\mathbf{M}$ with $\lambda_*(\xi') = \lambda_*(\xi)$, and $G(\xi)$ is the number classifiers ξ' in $\mathbf{M}$ with $\lambda_*(\xi') > \lambda_*(\xi)$.

We use these probabilities to exert pressure for more efficient covers by modifying the endogenous fitness scheme to use

$$\rho'(\xi) = \rho(\xi)\hbar(\xi)$$

when allocating resources to $\mathbf{A}$. This biases the allocation of resources in the desired way, making elements of a good cover more likely to reproduce. This enhanced version of the endogenous fitness scheme was tested on the 6-bit multiplexor problem. The same parameters were used as before, except that we used an increased value $\mathbf{R} = \pm 25,000$ to compensate for the change of scale caused by using the hypergeometric probabilities in the computation of ρ'. Figure 1 shows that this change does indeed improve performance.

Figure 3 shows that the performance improvements include a much better capability to generate efficient covers. The system reliably finds all of the maximally

```
(State = 0#11##) ==> 1 (8,1000.0)
(State = 001###) ==> 0 (17,-1000.0)
(State = 001###) ==> 1 (23,1000.0)
(State = 000###) ==> 0 (21,1000.0)
(State = 000###) ==> 1 (16,-1000.0)
(State = 01#0##) ==> 0 (28,1000.0)
(State = 01#0##) ==> 1 (19,-1000.0)
(State = 01#1##) ==> 0 (9,-1000.0)
(State = 01#1##) ==> 1 (12,1000.0)
(State = 10##1#) ==> 0 (24,-1000.0)
(State = 10##1#) ==> 1 (16,1000.0)
(State = 10##0#) ==> 0 (25,1000.0)
(State = 10##0#) ==> 1 (19,-1000.0)
(State = 11###1) ==> 0 (28,-1000.0)
(State = 11###1) ==> 1 (17,1000.0)
(State = 11###0) ==> 0 (20,1000.0)
(State = 11###0) ==> 1 (24,-1000.0)
```

Figure 3: Rules learned in the enhanced endogenous fitness scheme

general classifiers, and they have a strong presence in the population. Populations of this kind are comparable to what we expect from systems like XCS.

5 SUMMARY

The results presented here are modest and preliminary. However, they strongly suggest that endogenous fitness schemes are worth further investigation. While more experiments are needed to build our intuitions about how this approach can be used, the most important thing to do is formally analyze how the fluctuations in reservoir levels are translated into reproduction rates. That kind of understanding is vital if these schemes are ever to become reliable approaches for building classifier systems. Our current research efforts are focused on conducting that analysis.

† Acknowledgments

This research was funded by the MITRE Mission-Oriented Investigation and Experimentation (MOIE) research program. That support is gratefully acknowledged.

References

Booker, L. B. (1988). Classifier systems that learn internal world models. *Machine Learning, 3*, 161–192.

Booker, Lashon B. (1989). Triggered rule discovery in classifier systems. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 258–267). Fairfax, VA: Morgan Kaufmann.

Holland, J. H. (1976). Adaptation. In R. Rosen and F. M. Snell (Eds.), *Progress in theoretical biology* (Vol. 4). New York: Academic Press.

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.

Holland, J. H. (1994). Echoing emergence: Objectives, rough definitions, and speculations for ECHO-class models. In G. Cowan, D. Pines, and D. Melzner (Eds.) *Complexity: Metaphors, Models, and Reality.* Reading, MA: Addison-Wesley.

Horn, J., Goldberg, D. and Deb, K. (1994). Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation, 2(1)*, 37–66.

Smith, R. E., Forrest, S. and Perelson, A. S. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation, 1(2)*, 127–149.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation, 3(2)*, 149–175.