A Simple Accuracy-based Learning Classifier System

Larry Bull

Learning Classifier Systems Group Technical Report *UWELCSG03-005*

University of the West of England,

Bristol, BS16 1QY, U.K.

larry.bull@uwe.ac.uk

Learning Classifier Systems use evolutionary algorithms to facilitate rule-discovery, where rule fitness is traditionally payoff based and assigned under a sharing scheme. Most current research has shifted to the use of accuracy-based fitness, after the introduction of XCS, where rule fitness is based on a rule's ability to predict the expected payoff from its use. Whilst XCS has been shown to be extremely effective in a number of domains, its complexity can make it difficult to establish clear reasons for its behaviour. This paper presents a simple accuracy-based learning classifier system with which to explore aspects of accuracy-based fitness in general. The system is described and modelled, before being implemented and tested on the multiplexer task.

1. Introduction

Since its introduction Holland's Learning Classifier System (LCS) [Holland, 1976] has inspired much research into 'genetics-based' machine learning [Goldberg, 1989]. However, the developed system [Holland, 1986] was somewhat complex and experience found it difficult to realise the envisaged performance/behaviour [e.g., Wilson & Goldberg, 1989]. As a consequence, Wilson presented the 'zeroth-level' classifier system, ZCS [Wilson, 1994], which "keeps much of Holland's original framework but simplifies it to increase understandability and performance" [ibid.]. It has recently been shown that ZCS can perform optimally [Bull & Hurst, 2002] but "it would appear that the interaction between the rate of rule updates and the fitness sharing process is critical" [ibid.].

Most current research has made a shift away from Holland's original formalism, moving LCS much closer to the field of reinforcement learning [Sutton & Barto, 1998], after Wilson introduced XCS [Wilson, 1995]. XCS uses the accuracy of rules' predictions of expected payoff as their fitness. In this way a full map of the problem space is created, rather than the traditional search for only high payoff rules, with (potentially) maximally accurate generalizations over the state-action space [ibid.]. That is, XCS uses a genetic algorithm (GA)[Holland, 1975] to evolve generalizations over the space of possible state-action pairs with the aim of easing the use of such approaches in large problems, i.e. those with state-action combinations too numerous for an explicit entry for each (other techniques have been proposed - see [Sutton & Barto, 1998] for an overview). XCS can also avoid problematic 'overgeneral' rules which receive a high optimal payoff for some inputs but are sub-optimal for other, lower payoff, inputs. Since their average payoff is higher than that for the optimal rules in the latter case the overgenerals tend to displace them, leaving the LCS sub-optimal. However, the payoffs received by overgeneral rules typically have high variance (they are inaccurate predictors) and so have low fitness in XCS. Holland's LCS was shown to suffer due to such rules emerging [e.g., Dorigo 1993] although, when working effectively, fitness sharing can combat overgenerals [Bull & Hurst, 2002].

XCS has been shown to perform well in a number of domains [e.g., Lanzi et al., 2000] but how this is achieved is not well-understood, although some progress has been made [e.g., Butz et al., 2001]. In this paper, a simple accuracy-based LCS is presented which keeps much of Wilson's framework but simplifies it to increase understandability.

2. YCS: A Simple Accuracy-based Learning Classifier System

YCS is a Learning Classifier System without internal memory, where the rulebase consists of a number (N) of condition/action rules in which the condition is a string of characters from the usual ternary alphabet {0,1,#} and the action is represented by a binary string. Associated with each rule is a predicted payoff value p, a scalar which indicates the error (ε) in the rule's predicted payoff and an estimate of the average size of the niches (action sets - see below) in which that rule participates (α). The initial random population have these parameters initialized to 10.

On receipt of an input message, the rulebase is scanned, and any rule whose condition matches the message at each position is tagged as a member of the current match set [M]. An action is then chosen from those proposed by the members of the match set and all rules proposing the selected action form an action set [A]. The grouping of concurrently activated rules by action was used in the purely GA-based LS-1 [Smith, 1980] for action selection, and is used in ZCS and XCS for both action selection and reinforcement updates. A variety of action selection schemes are possible but a version of XCS's explore/exploit scheme will be used here. That is, on one cycle an action is chosen at random and on the following the action with highest average payoff is chosen deterministically.

In this paper the simplest case of immediate reward (payoff P) is considered. Reinforcement in YCS consists of updating the error, the niche size estimate and then the payoff estimate of each member of the current [A] using the Widrow-Hoff delta rule with learning rate β:

$$\varepsilon_j \leftarrow \varepsilon_j + \beta(|P - p_j| - \varepsilon_j) \quad (1)$$

$$\alpha_j \leftarrow \alpha_j + \beta(|[A]| - \alpha_j) \quad (2)$$

$$p_j \leftarrow p_j + \beta(P - p_j) \quad (3)$$

YCS employs two discovery mechanisms, a panmictic GA and a covering operator. On each time-step there is a probability g of GA invocation. When called, the GA uses roulette wheel selection to determine two parent rules based on the inverse of their error:

$$\text{fitness, } f_j = 1/(\varepsilon_j+1) \quad (4)$$

Offspring are produced via mutation (probability μ) and crossover (single point with probability χ), inheriting the parents' parameter values or their average if crossover is invoked. Replacement of existing members of the rulebase uses roulette wheel selection based on estimated niche size. If no rules match on a given time step, then a covering operator is used which creates a rule with the message as its condition (augmented with wildcards at the rate $p_{\#}$) and a random action, which then replaces an existing member of the rulebase in the usual way. The GA is not invoked on exploit trials.

Thus YCS represents a simple accuracy-based LCS which captures many of the key features of XCS: "[E]ach classifier maintains a prediction of expected payoff, but the classifier's fitness is *not* given by the prediction. Instead, the fitness is a separate number based on an inverse function of the classifier's average prediction error" [Wilson, 1995] and "[a] classifier's deletion probability is set proportional to the [niche] size estimate, which tends to make all [niches] have about the same size, so that classifier resources are allocated more or less equally to all niches" [ibid.].

The main difference between YCS and XCS is that the former, unlike the latter, has no mechanisms by which to form a maximally general mapping of the state-action space since it does not use a triggered niche GA nor any form of subsumption (see [Butz & Wilson, 2001] for a detailed description of XCS). However, those mechanisms, along with the others found in XCS, as will be shown, need not be considered as pre-requisites for the *effective* use of an accuracy-based fitness scheme.

The mechanisms of YCS are now modelled, in keeping with its philosophy, in a simple way.

3. A Simple Model of YCS

The evolutionary algorithm in YCS, and most LCS, is a steady-state GA. A simple steady-state GA without genetic operators can be expressed in the form:

$$n(k, t+1) = n(k, t) + n(k, t) R(k, t) - n(k, t) D(k, t) \quad (5)$$

where $n(k, t)$ refers to the number of individuals of structure type k in the population at time t, $R(k, t)$ refers to their probability of reproductive selection and $D(k, t)$ to their probability of deletion. Roulette-wheel selection is used in YCS, i.e., $R(k, t) = f(k, t)/f(K, t)$, where $f(k, t)$ is the fitness of individuals of type k (Equation 4) and $f(K, t)$ is the total population fitness. Replacement is proportional to action set size, i.e., $D(k, t) = \alpha(k, t)/\alpha(K, t)$.

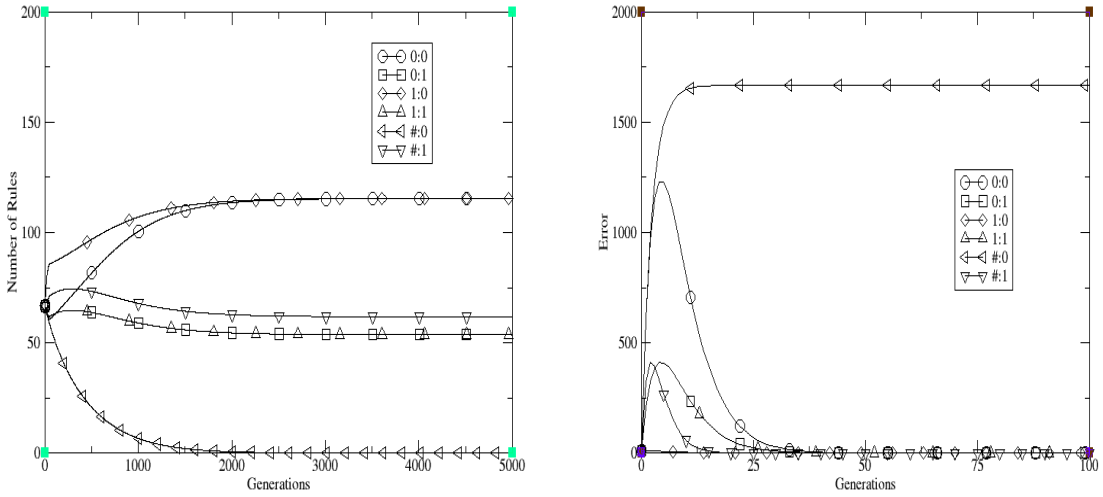
Table 1. Reward payoffs for the single-step task considered.

Input	Action	Payoff
1	1	1000
1	0	0
0	1	1000
0	0	3000
#	0	1500
#	1	1000

Table 1 shows the payoffs for the single-step task with a single-bit condition and single-bit action considered here. The last two entries in Table 1 show the expected payoff for the general rules, i.e., the predicted payoff of a general rule is the average of the payoffs it receives. It can be seen that under this scheme for input '1' the general rule #:0 has a higher predicted payoff than the correct rule 1:1; #:0 is an overgeneral rule which would cause sub-optimal performance. The progress of all six rules is examined here, with rulebase size N=400.

With equations of the general form shown in Equation 5 the expected proportions of each rule type in the next generation can be determined; by specifying the initial proportions of each rule in the population (N/6), it is possible to generate the trajectory of their proportions over succeeding generations. Note partial individuals are allowed and

hence it is in effect an infinite population model. The trajectory of their related parameters can also be generated. In the following it is assumed that both inputs are presented with equal frequency, that both actions are chosen with equal frequency and that the GA fires once every four cycles (i.e., always explore trials and g=0.25). The rules' parameters are updated according to Equations 1 to 3 on each cycle with β=0.2.



(a) (b)

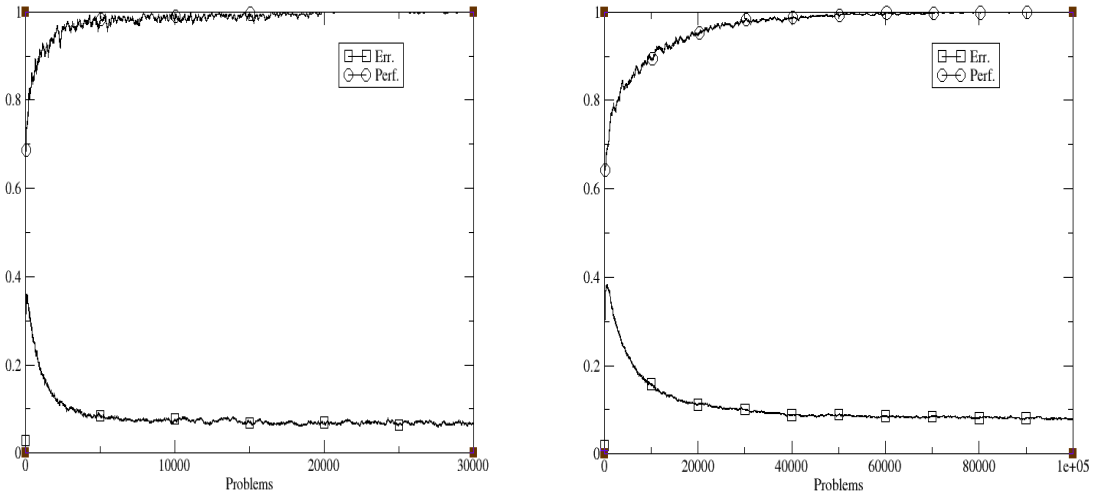
Figure 1. Behaviour of model YCS on the task in Table 1, showing numerosities (a) and errors (b).

Figure 1 shows the behaviour of the modelled YCS on the single-step task defined in Table 1. It can be seen that the inaccurate rule #:0 is rapidly squeezed out of the population (Figure 1(a)) since it has a high error (Figure 1(b)). The two accurate rules with action '0' gain a slightly larger fraction of the rulebase than those with action '1' but the system is roughly balanced. That is, the rule replacement scheme based on action set size appears to work effectively here. Figure 1(b) shows the errors of the rules over the first 100 GA events (400 system cycles). It can be seen that, apart from the overgeneral rule #:0, errors rapidly drop to zero after initial adjustments. Therefore the simple accuracy-based fitness scheme results in a rulebase capable of optimal performance under the exploit action selection scheme described above.

4. YCS on the Multiplexer Problem

YCS has been implemented and investigated using versions of the well-known multiplexer task. These Boolean functions are defined for binary strings of length $l = k + 2^k$ under which the first k bits index into the remaining 2^k bits, returning the value of the indexed bit. A correct classification results in a payoff of 1000, otherwise 0.

Figure 2(a) shows the performance of YCS, as described in Section 2, on the 6-bit multiplexer problem using the same parameters as in Section 3, with $p_{\#}$=0.6, χ=0.5 and μ=0.01. After [Wilson, 1995], performance from exploit trials only is recorded (fraction of correct responses are shown), using a 50-point running average, averaged over ten runs. It can be seen that YCS is capable of optimal performance and that the average error (shown divided by the payoff range) of the rules drops below 10% of the payoff range. Figure 2(b) shows the performance of the same system on the 11-bit multiplexer with N=2000. Again, it can be seen that YCS achieves optimal performance. Hence, as indicated by the model in the previous section, YCS is an effective accuracy-based LCS despite its relative simplicity.



(a) (b)

Figure 2. Performance of YCS on the multiplexer task, 6-bit (a) and 11-bit (b) versions.

5. Conclusion

The use of accuracy-based fitness in Learning Classifier Systems has a long history. In the first implementation of LCS, rule fitness was viewed as a predictor of expected future payoff [Holland & Reitman, 1978]. Payoff was given only to those rules which predicted a reward no greater than the actual value received from the environment - "to reflect their accuracy in anticipating this reward" [ibid.]. Booker [1982] factored accuracy into rule fitness and used niche size for replacement. Grefenstette et al. [1990] also factored accuracy into fitness, as did Riolo [1991] in his version of Holland's system for exploiting lookahead and latent learning (see [Butz & Stolzmann, 2001] for a related approach). Frey and Slate [1991] used accuracy as rule fitness, recording the number of times a rule correctly classified an input. Bull [2002] has presented a related, but more general, approach which uses a strict accuracy criterion and fitness sharing to balance rulebase resources. Booker's [2000] ecology-based system can also been seen to use accuracy. However, it is Wilson's [1995] fundamentally accuracy-based XCS which has demonstrated the potential of the scheme.

This paper has presented a simple accuracy-based LCS which draws heavily on Wilson's XCS. The aim of this work is to enable a greater understanding of accuracy-based fitness in general, i.e., outside of the many other mechanisms contained within XCS. Future work will examine the use of YCS with other mechanisms incoporated, in more complex single-step tasks and in multi-step tasks.

Acknowledgement

Thanks to the members of the LCS Group at UWE for many useful discussions during this work.

References

Booker, L.B. (1982) *Intelligent Behaviour as an Adaptation to the Task Environment*. Ph.D. Disertation, University of Michigan.

Booker, L.B. (2000) Do we Really Need to Estimate Rule Utilities in Classifier Systems? In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Learning Classifier Systems: From Foundations to Applications*. Spinger, pp125-142.

Bull, L. (2002) Lookahead and Latent Learning in ZCS. In W.B.Langdon, E.Cantu-Paz, K.Mathias, R. Roy, D.Davis, R. Poli, K.Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke & N.Jonoska (eds) *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference.* Morgan Kaufmann, pp897-904.

Bull, L. & Hurst, J. (2002) ZCS Redux. *Evolutionary Computation* 10(2): 185-205.

Butz, M. & Stolzmann, W. (2002) An Algorithmic Description of ACS2. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Advances in Learning Classifier Systems: IWLCS 2001*. Springer, pp211-230.

Butz, M. & Wilson, S.W. (2001) An Algorithmic Description of XCS. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Advances in Learning Classifier Systems: IWLCS 2000*. Springer, pp253-272.

Dorigo, M. (1993) Genetic and Non-Genetic Operators in ALECSYS. *Evolutionary Computation* 1(2):151-164.

Frey, P.W. & Slate, D.J. (1991) Letter Recognition using Holland-style Adaptive Classifiers. *Machine Learning* 6: 161-182.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley.

Grefenstette, J.J., Ramsey, C.L. & Schultz, A.C. (1990) Learning Sequential Decision Rules using Simulation Models and Competition. *Machine Learning* 5: 355-381.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Holland, J.H. (1976) Adaptation. In R. Rosen & F.M. Snell (eds) *Progress in Theoretical Biology, 4.* Plenum.

Holland, J.H. (1986) Escaping Brittleness. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (eds) *Machine Learning: An Artificial Intelligence Approach, 2*. Morgan Kauffman, pp48-78.

Holland, J.H. & Reitman, J.S. (1978) Cognitive Systems Based on Adaptive Algorithms. In D.A. Waterman & F Hayes-Roth (eds) *Pattern-directed Inference Systems*. Academic Press, pp313-331.

Lanzi, P-L., Stolzmann, W. & Wilson, S.W. (2000)(eds) *Learning Classifier Systems: From Foundations to Applications*. Spinger.

Smith, S.F. (1980) *A Learning System Based on Genetic Adaptive Algorithms*. Ph.D. Dissertation, University of Pittsburgh.

Sutton, R.S. & Barto, A.G. (1998) *Reinforcement Learning*. MIT Press.

Wilson, S.W. (1994) ZCS: A Zeroth-level Classifier System. *Evolutionary Computation* 2(1):1-18.

Wilson, S.W. (1995) Classifier Fitness Based on Accuracy. *Evolutionary Computation* 3(2):149-177

Wilson, S.W. & Goldberg, D.E. (1989) A Critical Review of Classifier Systems. In J.D. Schaffer (ed) *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, pp244-255.