High Classification Accuracy Does Not Imply Effective Genetic Search

Tim Kovacs[1] and Manfred Kerber[2]

[1] University of Bristol, Bristol BS8 1UB, U.K.
kovacs@cs.bris.ac.uk
http://www.cs.bris.ac.uk/~kovacs
[2] University of Birmingham, Birmingham B15 2TT, U.K.
M.Kerber@cs.bham.ac.uk
http://www.cs.bham.ac.uk/~mmk

Abstract. Learning classifier systems, their parameterisation, and their rule discovery systems have often been evaluated by measuring classification accuracy on small Boolean functions. We demonstrate that by restricting the rule set to the initial random population high classification accuracy can still be achieved, and that relatively small functions require few rules. We argue this demonstrates that high classification accuracy on small functions is not evidence of effective rule discovery. However, we argue that small functions can nonetheless be used to evaluate rule discovery when a certain more powerful type of metric is used.

1 Introduction

Much research on Learning Classifier Systems (LCS) has made use of small artificial data sets to evaluate alternative systems, mechanisms, and parameter settings. Of these data sets, the venerable 6 multiplexer (a 6-bit Boolean function) is the most widely used test in the LCS literature [36,33,37,6,13,5,35,25, 11,14,15,10,39,16,18,19,40,9,20,3,8,21]. It has also been used with other machine learning systems including neural networks [4,2,17,38,5,34], decision trees [30,29], and the GPAC algorithm [28]. See [25] for a review of some of the earlier work using the multiplexer. A number of studies have looked at larger multiplexer functions, e.g. [39,40,22], up to the 70-bit multiplexer [7].

Although some authors have referred to the 6 multiplexer as a difficult task, section 3 demonstrates that the XCS classifier system with a fixed set of random rules, of the same number as standardly used by the XCS for this task, reliably achieves 100% classification accuracy. That is, we discontinue genetic search in XCS (except for covering) after the initial random rule population has been generated. We refer to this algorithm as XCS-NGA (XCS-No Genetic Algorithm). This result clearly demonstrates that good classification accuracy on this task is not evidence of the efficacy of genetic search. (We note that rule parameters such as fitness and prediction are updated as usual by the credit assignment system, and that we are thus dealing with fixed randomly defined regions of the input space rather than random classification of inputs.)

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 785–796, 2004.

The same holds for other small data sets. We demonstrate 98% classification accuracy with XCS-NGA on the 11-bit multiplexer, although this requires four times as many rules as normally used by XCS. We suggest any degree of accuracy can be achieved on any task, given sufficient random rules. We do not suggest this is a practical approach as the number of rules required scales poorly. However, we do suggest that this is a useful measure of the difficulty of a given task, and one which can potentially demonstrate the invalidity of conclusions drawn from experiments with smaller tasks, for example those with the 6 multiplexer.

Despite this criticism of the use of small tasks, we argue that many conclusions drawn from them are valid. For example, comparisons based on small tasks can demonstrate performance differences between alternative mechanisms and parameterisations. We demonstrate this in section 3.3 by comparing the performance of XCS with and without initial populations on the 6 multiplexer.

We further demonstrate in section 4 that an alternative measure of adaptation can distinguish between XCS and XCS-NGA on the 6 multiplexer, and argue that the use of this metric increases the utility of small tests.

2 Method

2.1 XCS

For our experiments we will use XCS [39], which is currently the most widely used classifier system, and which has shown good results on data mining tasks (e.g. [31,12]). XCS introduced a number of innovations, foremost among them its accuracy-based fitness under which rule fitness is related to its classification accuracy and not the magnitude of the reward it receives as in earlier systems. For lack of space we do not include the details of the XCS updates, but suffice it to say that XCS evaluates the prediction and fitness of each rule. Prediction is, for concept learning tasks such as those we study here, an estimate of the proportion of inputs matched by the rule which belong to the positive class. Prediction is used in conflict resolution, when matching rules perform a weighted vote on the classification of a data point. Accuracy is a measure of the consistency of prediction. Rules with prediction near the maximum or minimum have high fitness. Higher fitness rules are allocated more reproductive opportunities by the genetic algorithm in XCS, and fitness is also factored into the classification vote.

For our experiments we use if-then rules whose conditions are terms in Disjunctive Normal Form. Specifically, we use the ternary representation widely used with classifier systems, in which rule conditions are drawn from $\{0, 1, \#\}$ and rule actions (classifications) are drawn from $\{0, 1\}$. Inputs to the system are also drawn from $\{0, 1\}$. A rule's condition c matches an environmental input m if for each character m_i the character in the corresponding position c_i is identical or the wildcard (#). For example, the condition $0\,0\,\#$ matches two inputs: 000 and 001. The wildcard is the means by which rules generalise over environmental states; the more #s a rule contains the more general it is. *Overgeneral* rules are those which misclassify some of the inputs they match. Since actions do not contain wildcards the system cannot generalise over them.

If no rule matches the current input, XCS's *covering* mechanism is triggered. This mechanism takes the current input and with probability $P_\#$ for each bit flips it to a #, and uses this as the condition for a new rule with a random classification. XCS may or may not use an initial population of random rules whose conditions are generated with $P_\#$ and equiprobable 0s and 1s. The covering mechanism is used regardless of whether an initial population is used, but, when $P_\#$ is not very close to 0, covering is triggered only sparingly and typically only at the outset of the experiment, even in the absence of an initial population.

2.2 XCS-NGA

Our procedure for learning with random rules, XCS-NGA, uses XCS modified so that genetic search does not operate on the initial rule population. In all other respects, XCS-NGA functions as XCS. The adaptive power of this approach lies in the XCS updates which estimate the prediction and fitness of rules, and weight classification votes on these two values. In section 3.1 we give some intuition as to why this approach can be effective.

We could attempt to improve XCS-NGA by restricting the generality of rules found to be overgeneral, or simply deleting them. However, our aim is not to propose a practical learning technique but rather to provide a baseline against which to evaluate other methods.

We note that in experimental results presented later we will quote a certain number of random rules having been used. In some experiments, some additional rules will have been generated by covering. In these cases, the same number of initial random rules will have been removed from the population in order to make room for the rules generated by covering. In most experiments covering does not occur, and when it does (typically when the rule set is small) it is not triggered many times.

2.3 The Multiplexer Tests

The 6 multiplexer is one of a family of Boolean multiplexer functions defined for strings of length $L = k + 2^k$ where k is an integer > 0. The series begins $L = 3, 6, 11, 20, 37, 70, 135, 264, 521 \ldots$. The first k bits are used to encode an address into the remaining 2^k bits, and the value of the function is the value of the addressed bit. In the 6 multiplexer ($k = 2, L = 6$), the input to the system consists of a string of six binary digits, of which the first $k = 2$ bits (the address) represent an index into the remaining $2^k = 4$ bits (the data). For example, the value of 101101 is 0 as the first two bits 10 represent the index 2 (in base ten) which is the third bit following the address. Similarly, the value of 001000 is 1 as the 0th bit after the address is indexed.

To use the 6 multiplexer as a test, on each time step we generate a random binary string of 6 digits which we present as input to the LCS. The LCS responds with either a 0 or 1, and receives a high reward (1000) if its output is that of the multiplexer function on the same string, and a low reward (0) otherwise.

2.4 Statistics

Classification Accuracy. We make use of Wilson's explore/exploit framework [39], in which training and testing interleave, so the learner is evaluated as it is learning, rather than after it has been trained. Specifically, on each time step we alternate between explore and exploit modes. In the former we select an action at random from among those advocated by the set of matching rules. In the latter we select the action most strongly advocated by the matching rules. We record statistics only on those time steps in which we exploit (exploit trials).

Wilson defines a measure of performance which he refers to simply as "performance" [39]. Performance is defined as a moving average of the proportion of the last n trials in which the system has responded with the correct action, where n is customarily 50. That is, on each time step, we determine the proportion of the last n time steps on which the LCS has taken the correct action. The performance curve is scaled so that when the system has acted correctly on all of the last 50 time steps it reaches the top of the figure, and when it has acted incorrectly on all these time steps it reaches the bottom of the figure.

Macroclassifiers. In addition to performance, on each exploit trial we monitor the number of *macro*classifiers in the population. These are rules with a numerosity parameter indicating the number of identical virtual rules represented by the macroclassifier. The macroclassifier curves gives us an indication of the diversity in the rule population and the extent to which it has found and converged on useful general rules and hence a compact representation of the solution. When an initial population is used the macroclassifier curve starts a little below the specified population size limit since a few duplicate rules are likely to have been generated. This curve can at most reach the population size limit, which would occur when each rule in the population has a unique condition/action pair. In the figures shown later, the number of macroclassifiers is divided by 1000 in order to display it simultaneously with other curves.

2.5 Parameter Settings

For the 6 multiplexer the standard XCS parameter settings from [39] were used: subsumption threshold $\theta_{sub} = 20$, Genetic Algorithm (GA) threshold $\theta_{GA} = 25$, covering threshold $\theta_{mna} = 1$, low-fitness deletion threshold $\delta = 0.1$, population size limit $N = 400$, learning rate $\beta = 0.2$, accuracy falloff rate $\alpha = 0.1$, accuracy criterion $\varepsilon_o = 0.01$, crossover rate $\chi = 0.8$, mutation rate $\mu = 0.04$. Hash probability $P_{\#} = 0.3$ rather than the standard 0.33 as a study of different values (not shown) was performed with hash probabilities at regular intervals, and the results shown are a subset of this study. GA subsumption was used but not action set subsumption. The original accuracy calculation was used [39]. Rules were deleted as in [20] with a delay of $\theta_{del} = 25$, and mutated bits had equiprobable outcomes. Initial random populations were used except as noted.

Parameter settings for the 11 multiplexer differed only in having a population size of 800 and hash probability of 0.4, to compensate for the larger search space.

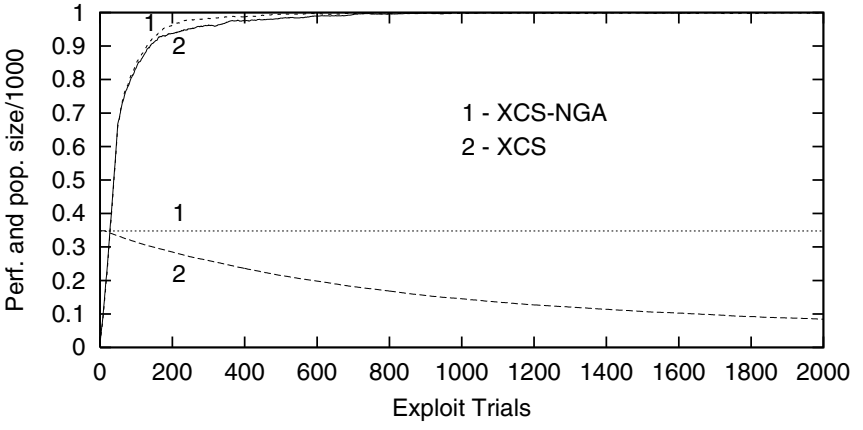

Fig. 1. XCS and XCS-NGA compared on the 6 multiplexer. The upper two curves show performance and the lower two show the population size in macroclassifiers.

3 The Difficulty of Small Boolean Functions

In this section we evaluate XCS and XCS-NGA empirically on the 6 and 11-bit multiplexer functions and discuss implications of our findings. Fig. 1 compares XCS and XCS-NGA on the 6 multiplexer. Curves are averages of 100 runs. The upper two curves show performance and the lower two show the population size in macroclassifiers (divided by 1000). Although 400 initial rules were generated for each system, both population size curves start somewhat below this as the curves show macroclassifiers, and some duplicate rules were generated. Both systems reach 100% performance, and, perhaps surprisingly, XCS-NGA does so first. We note that random guessing of class has an expected performance of 50% on multiplexer tasks, and, given the even class distribution, guessing the majority class of previously seen inputs will perform somewhat worse than random [32].

Fig. 2 repeats the comparison using the 11 multiplexer. Curves are an average of 50 runs. XCS-NGA was evaluated with 800 and 3200 rules. In both cases XCS-NGA initially outperformed XCS. XCS-NGA with 800 random rules ultimately achieve approximately 86% classification accuracy and with 3200 rules reached approximately 98%, while XCS eventually reaches 100%. (A larger number of random rules should do even better, but in a sense 98% is high enough: using a method with disjoint training and testing data sets, overfitting will occur on many data sets by the time 98% performance is reached.)

3.1 How XCS-NGA Achieves High Accuracy

Suppose we have a space of data points to be categorised. XCS uses a generate-and-test approach to classification, which entails two problems: i) rule discovery

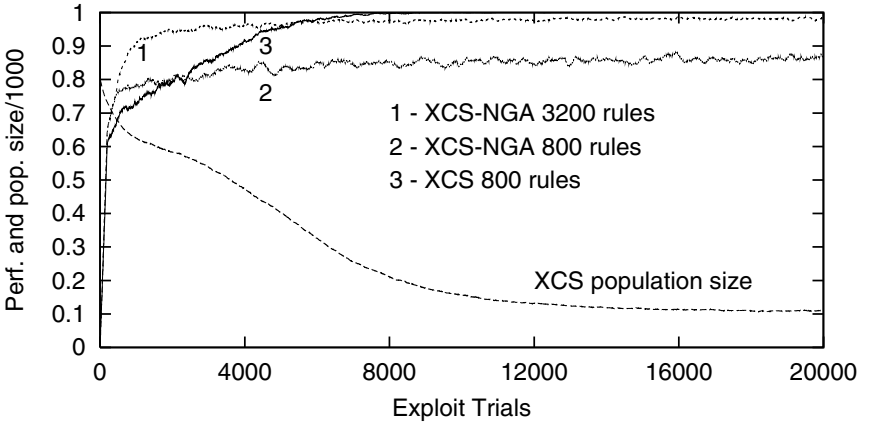

Fig. 2. XCS and XCS-NGA compared on the 11 multiplexer.

and ii) credit assignment. Specifically, XCS addresses problem i) using a GA to generate fitter rules (regions in the data space), each with an associated class label. Problem ii) is that of evaluating rule fitness such that more general rules and rules with higher classification accuracy are fitter. Essentially, rules must be found which capture many positive data points and few negative ones (or vice versa). XCS classifies data points by a vote among the rules which match it, with each vote weighted both by the rule's fitness. In this way, a point matched by a low-accuracy rule and a high-accuracy rule is given the classification of the high-accuracy rule.

In XCS, the rules (region shapes and sizes) are adapted by the genetic algorithm. XCS-NGA lacks a GA and its region shapes and sizes do not change; only the classification made through voting may change. XCS-NGA relies on there being enough rules to adequately solve problem i) (rule discovery) by chance. Of the randomly generated rules, those with low classification accuracy are assigned low weights and have less influence in the classification vote than higher accuracy rules. Roughly speaking, XCS-NGA's approach is to generate many random rules and ignore those which happen to have low accuracy. The number of random rules needed for high classification accuracy on small multiplexers is low because there are relatively few data points and clustering them into regions of the same class is easy (using our chosen language).

The difficulty of the rule discovery problem depends on the Kolmogorov complexity[1] of the data set. There is considerable variability in the Kolmogorov complexity of functions of the same length and representation. For example, with the language we have used the 6-bit parity functions are much more complex than the 6 multiplexer, which in turn is considerably more complex than 6-bit constant functions. Elsewhere, we have demonstrated a strong correlation between the size

[1] In simple terms, the shortest possible representation in a given formalism [24].

of the minimal representation of these functions and their difficulty for XCS [23]. One consequence is that even successful solution by XCS of a large multiplexer, such as the 70-bit multiplexer [7], does not mean that XCS can solve all 70-bit functions with comparable effort; quite the opposite. We hypothesise that the difficulty of a function for XCS-NGA will also correlate with the minimal number of rules needed to represent the function in a particular language.

XCS-NGA is related to a number of other machine learning algorithms. For example, CMAC function approximators [1] adapt the weight of each region in each of multiple partitions of the input space. Partitions may be regular or generated at random, and XCS-NGA differs essentially only in the details of how regions are formed. XCS-NGA is also very similar to the Weighted-Majority algorithm [27], which enumerates all possible concepts and weights them according to their consistency with the training data. They differ in that XCS-NGA generates only a random subset of possible concepts.

3.2 Why XCS-NGA Initially Outperforms XCS

It is not clear why XCS-NGA initially outperform XCS, but it may be that XCS is deleting overgeneral rules which have some value; overgenerals can advocate the correct action for the great majority of inputs they match. In both XCS and XCS-NGA, overgeneral rules have low accuracy and hence low weight in action selection, but nonetheless may have some effect. In XCS, however, low accuracy results in low fitness and greater likelihood of deletion under the genetic algorithm, and once deleted rules have no effect. Further study of this phenomenon is warranted, and perhaps improved performance in XCS can be obtained by allowing it to retain overgeneral rules when no accurate rule matches an input, or by delaying the application of the GA to the initial population until it has been better evaluated.

3.3 Implications of High Accuracy of XCS-NGA

Although XCS outperforms XCS-NGA on the 11 multiplexer, it seems likely that XCS-NGA with a large enough set of random rules would also reach 100% performance on this function, or indeed any function.

Although we have shown that good performance on the 6 multiplexer with 400 rules does not demonstrate effective genetic search in a classifier system, we do not claim that the 6 multiplexer is without uses. For example, in section 2.1 we noted that XCS may either use an initial population of rules or rely entirely on covering to generate rules. Fig. 3 compares XCS with and without an initial population of 400 rules on the 6 multiplexer, and clearly shows the performance advantage which occurs with an initial population. Consequently, we argue that only those studies which claim effective genetic search based on results with small functions are demonstrated invalid by our results with XCS-NGA.

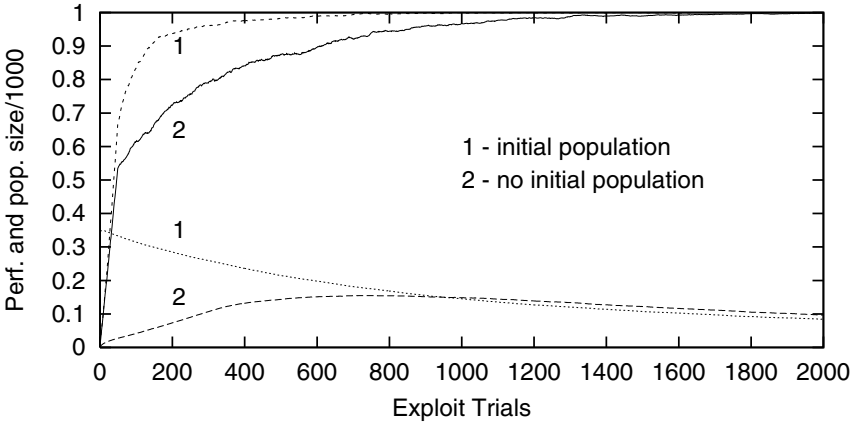

Fig. 3. XCS with and without an initial population on the 6 multiplexer.

4 A More Powerful Metric for Evaluating Genetic Search

High classification accuracy is usually regarded as the primary goal of a concept learning system. (Another goal might be human readability.) However, from the point of view of a researcher engaged in engineering better concept learning systems, classification accuracy is not a goal in itself, but just an indication of the relative merit of alternative mechanisms and parameterisations. In this context, classifier systems researchers often interpret good classification accuracy as an indication of effective genetic search for good classification rules. However, our demonstration in section 3 of the high classification accuracy achievable with XCS-NGA on the 6 and 11 multiplexers indicates that this interpretation is not justified when the number of rules is high relative to the size of the problem. This suggests that in order to evaluate the efficacy of genetic search good classification performance is required with a small number of rules on a large problem. Unfortunately, this approach requires evaluation of "large" and "small" in the context of a particular learning system, and running experiments on large problems is computationally expensive. In this section we demonstrate use of a metric which provides an alternative to large problems. Experiments with this metric clearly distinguish the efficacy of genetic search as opposed to random rules on the 6 multiplexer.

Using the rule language of section 2.1 the most compact description of the 6 multiplexer is a set of 8 rules, each as general as it can be without being overgeneral. Because XCS assigns fitness based on rule accuracy it actually finds each rule and its complement – the rule with the same condition but complementary action. This forms a set of 16 rules referred to as the *optimal solution* due to the optimal generality of each rule, and the minimality of the set.

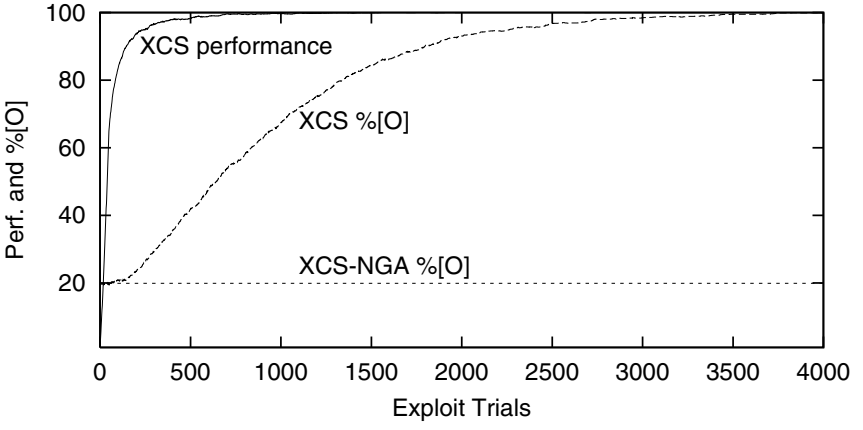

Fig. 4. Proportion of the optimal solution on the 6 multiplexer.

The proportion of the optimal solution in the rule population on a given time step, denoted %[O], has been used as a measure of the progress of genetic search. In [20] it was shown to have greater discriminatory power than the performance metric of section 2.4, and we will show that it can discriminate between XCS and XCS-NGA on the 6 multiplexer. Fig. 4 shows the performance of XCS and the %[O] of both XCS and XCS-NGA on this task. Curves are averages of 100 runs. While XCS-NGA contains a fixed proportion of approximately 20% of the optimal population, the proportion of this set grows in XCS until it reaches 100% (indicating all 16 rules occur in the population). Similarly, Fig. 5 compares %[O] for XCS and XCS-NGA on the 11 multiplexer, averaged over 50 runs. The size of the optimal solution for this function (including complementary rules) is 32. In this case XCS-NGA contains a much smaller proportion of the optimal set than in the 6 multiplexer because on the 11 multiplexer the optimal rules form a smaller proportion of the set of all rules.

Clearly this metric is better able to discern the progress of genetic search than the performance metric. We argue that using this metric extends the utility of small tests. However, we note that, as discussed in [22], %[O] has disadvantages including the need to compute the optimal solution in advance, the computational expense of evaluating it, and the complication that some functions have multiple optimal solutions (alternative minimal solution sets). These features make it difficult or impossible to apply %[O] to some tasks. In such cases measuring the population size in macroclassifiers or plotting mean rule generality can somewhat compensate for the lack of %[O], as decline of the former and rise of the latter both imply effective genetic search. Finally, replacing the GA with an iterative random rule generator would provide a baseline against which to compare genetic search.

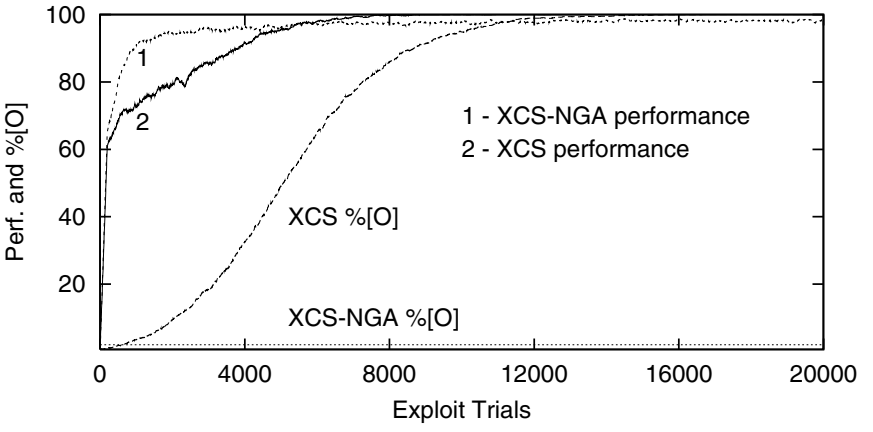

Fig. 5. Proportion of the optimal solution on the 11 multiplexer.

5 Conclusion

With XCS-NGA we have demonstrated that adapting the prediction and fitness of a fixed set of random rules reliably achieves 100% classification accuracy on the 6 multiplexer even when we only use as many rules as are standardly used to solve this task with XCS. This illustrates the danger of interpreting good classification accuracy on small tasks as indicative of successful genetic search. We have demonstrated the very high classification accuracy of XCS-NGA on the 11 multiplexer, and suggest that arbitrarily high accuracy can be achieved on any concept learning task given enough random rules. Given these results, we suggest that XCS-NGA can provide a useful baseline for classification accuracy.

We have demonstrated that, in contrast to performance, an alternative metric based on the proportion of the optimal solution present in the rule population clearly distinguishes the performance of fixed random rules and genetic search.

As a final point, we note that the use of small tasks is limited neither to classifier systems nor to artificial data sets. Although most data sets in the very widely used UCI repository of machine learning data sets [26] have larger attribute spaces than the 11-multiplexer studied here, many have fewer data points. Consequently a reasonable number of random rules seems likely to perform well on those data sets just as they do on those tested here.

References

1. J. S. Albus. A new approach to manipulator control: the cerebellar model articulation controller (CMAC). *Journal of dynamic systems, measurement and control*, 97(3), 1975.
2. C. W. Anderson. *Learning and Problem solving with multilayer connectionist systems*. PhD thesis, University of Massachussetts, Amherst, MA, USA, 1986.

3. Alwyn Barry. *XCS Performance and Population Structure within Multiple-Step Environments*. PhD thesis, Queens University Belfast, 2000.
4. A. G. Barto, P. Anandan, and C. W. Anderson. Cooperativity in networks of pattern recognizing stochastic learning automata. In *Proceedings of the Fourth Yale Workshop on Applications of Adaptive Systems Theory*, pages 85–90, 1985.
5. Pierre Bonelli, Alexandre Parodi, Sandip Sen, and Stewart Wilson. NEWBOOLE: A Fast GBML System. In *International Conference on Machine Learning*, pages 153–159, San Mateo, California, 1990. Morgan Kaufmann.
6. Lashon B. Booker. Triggered rule discovery in classifier systems. In J. David Schaffer, editor, *Proc. 3rd International Conference on Genetic Algorithms (ICGA-89)*, pages 265–274, George Mason University, June 1989. Morgan Kaufmann.
7. Martin Butz, Tim Kovacs, Pier Luca Lanzi, and Stewart W. Wilson. Toward a theory of generalization and learning in xcs. *To appear in the IEEE Transactions on Evolutionary Computation*, 2004.
8. Martin V. Butz, David E. Goldberg, and Wolfgang Stolzmann. Investigating Generalization in the Anticipatory Classifier System. In *Proceedings of Parallel Problem Solving from Nature (PPSN VI)*, 2000. Also technical report 2000014 of the Illinois Genetic Algorithms Laboratory.
9. Henry Brown Cribbs III and Robert E. Smith. What Can I do with a Learning Classifier System? In C. Karr and L. M. Freeman, editors, *Industrial Applications of Genetic Algorithms*, pages 299–320. CRC Press, 1998.
10. Bart de Boer. Classifier Systems: a useful approach to machine learning? Master's thesis, Leiden University, 1994.
11. Kenneth A. De Jong and Willliam M. Spears. Learning Concept Classification Rules Using Genetic Algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 651–656, Sidney, Australia, 1991.
12. Phillip William Dixon, David W. Corne, and Martin John Oates. A preliminary investigation of modified xcs as a generic data mining tool. In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Advances in Learning Classifier Systems*, volume 2321 of *LNAI*, pages 133–150. Springer-Verlag, Berlin, 2002.
13. David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.
14. David Greene and Stephen Smith. COGIN: Symbolic induction using genetic algorithms. In *Proceedings 10th National Conference on Artificial Intelligence*, pages 111–116. Morgan Kaufmann, 1992.
15. David Greene and Stephen Smith. Using Coverage as a Model Building Constraint in Learning Classifier Systems. *Evolutionary Computation*, 2(1):67–91, 1994.
16. John H. Holmes. *Evolution-Assisted Discovery of Sentinel Features in Epidemiologic Surveillance*. PhD thesis, Drexel University, 1996.
17. R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.
18. Tim Kovacs. Evolving Optimal Populations with XCS Classifier Systems. Master's thesis, University of Birmingham, 1996.
19. Tim Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. In Roy, Chawdhry, and Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer–Verlag, London, 1997.
20. Tim Kovacs. Deletion schemes for classifier systems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 329–336. Morgan Kaufmann, 1999.

21. Tim Kovacs. What should a classifier system learn? In *Proc. of the 2001 Congress on Evolutionary Computation (CEC01)*, pages 775–782. IEEE Press, 2001.
22. Tim Kovacs. *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Springer, 2004.
23. Tim Kovacs and Manfred Kerber. What makes a problem hard for XCS? In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Advances in Learning Classifier Systems*, LNAI 1996, pages 80–99. Springer–Verlag, 2001.
24. Ming Li and Paul Vitányi. An Introduction to Kolmogorov Complexity and Its Applications. Springer, 1997.
25. Gunar E. Liepins and Lori A. Wang. Classifier System Learning of Boolean Concepts. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pages 318–323, San Mateo, CA, 1991. Morgan Kaufmann Publishers.
26. Christopher Merz and P. M. Murphy. UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html Irvine, CA: University of California, Department of Information and Computer Science, 1997.
27. Tom M. Mitchell. *Machine Learning*. McGraw–Hill, 1997.
28. E. M. Oblow. Implementing Valiant's Learnability Theory using Random Sets. Technical Report ORNL/TM-11512R, Oak Ridge National Laboratory, 1990.
29. G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–100, 1990.
30. J. R. Quinlan. An empirical comparison of genetic and decision-tree classifiers. In *Proc. of the Fifth Int. Machine Learning Conference*, pages 135–141, 1988.
31. Shaun Saxon and Alwyn Barry. XCS and the Monk's Problems. In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *LNAI*, pages 223–242, Berlin, 2000. Springer-Verlag.
32. Cullen Schaffer. A conservation law for generalization performance. In Haym Hirsh and Willian W. Cohen, editors, *Machine Learning: Proc. of the Eleventh International Conference*, pages 259–265, San Francisco, CA, 1994. Morgan Kaufmann.
33. Sandip Sen. Classifier system learning of multiplexer function. The University of Alabama, Tuscaloosa, Alabama. Class Project, 1988.
34. Robert E. Smith and H. Brown Cribbs. Is a Learning Classifier System a Type of Neural Network? *Evolutionary Computation*, 2(1):19–36, 1994.
35. L. A. Wang. Classifier System Learning of the Boolean Multiplexer Function. Master's thesis, Computer Science Dept., University of Tennessee, Knoxville, 1990.
36. Stewart W. Wilson. Classifier Systems and the Animat Problem. *Machine Learning*, 2:199–228, 1987.
37. Stewart W. Wilson. Bid competition and specificity reconsidered. *Complex Systems*, 2:705–723, 1989.
38. Stewart W. Wilson. Perceptron Redux: Emergence of Structure. *Physica D*, pages 249–256, 1990.
39. Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
40. Stewart W. Wilson. Generalization in the XCS classifier system. In John Koza et al., editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998.