MILCS: A Mutual Information Learning Classifier System

R. E. Smith
University College London
Department of Computer Science
+44 777 185 2565
robert.elliott.smith@gmail.com

Max Kun Jiang
University College London
Department of Computer Science
+44 782 876 1996
m.jiang@cs.ucl.ac.uk

ABSTRACT
This paper introduces a new variety of learning classifier system (LCS), called MILCS, which utilizes mutual information as fitness feedback. Unlike most LCSs, MILCS is specifically designed for supervised learning. MILCS's design draws on an analogy to the structural learning approach of cascade correlation networks. We present preliminary results, and contrast them to results from XCS. We discuss the explanatory power of the resulting rule sets, and introduce a new technique for visualizing explanatory power. Final comments include future directions for this research, including investigations in neural networks and other systems.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning: Induction

General Terms
Algorithms

Keywords
Learning classifier systems, evolutionary computation, structural learning, supervised learning, cascade correlation, information theory, mutual information, visualization, explanatory power, rule learning.

1. INTRODUCTION
This paper presents a new form of learning classifier system (LCS) [6][8] that uses mutual information [11][12] as its primary fitness feedback, in supervised learning settings. This system is called the mutual information learning classifier system (MILCS, pronounced "my LCS"). In addition to drawing on current LCS research and information theoretic concerns, the system draws on an analogy to cascade correlation neural networks (CCNs) [5] in its design.

The following sections describe these inspirations and the general design of MILCS. Afterwards, we discuss preliminary results, with comparison to XCS. In this comparison, we are concerned not only with accuracy, generalization, and required computational time, but also with explanatory power of the resulting rule sets. Since explanatory power (in essence, the human understandability of machine-learned concepts) is an abstract concept, we introduce a new technique for visualizing results.

The preliminary results presented are promising, and in final comments we discuss future directions for this research. This clearly includes further research into MILCS performance, but also includes extension of the ideas involved to neural networks, and possibly other systems.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

2. LCSs and CCNs: An Analogy
To introduce the ideas in MILCS, it is first useful to consider the similarities and differences between LCSs and CCNs. This is done in several stages in the following subsections.

2.1 Parameter versus Structural Learning
In general, machine learning approaches can be broken into two components: the learning of continuous-valued parameters of a solution representation, and the learning of discrete connections between elements of the solution representation. The terms "parameter" and "structural" learning are borrowed from the Bayesian Network community [7], where the former is the learning of probabilities of events, conditioned on one another (the probabilities that reside within Bayesian Network nodes) and the latter is the learning of which events are conditioned on which (the links in the Bayesian Network). It is generally acknowledged that the latter is more difficult than the former. Also, the latter has a profound effect on the computational complexity of the former: the number of probabilities one must learn goes up exponentially with the number of links. Moreover, structural learning is also associated with generalization, parsimony, and explanatory power: fewer discrete connections make for a more understandable representation of a solution.

2.2 CCN and Structural Learning
It is also interesting to note that neural networks do not, in general, employ sophisticated techniques for structural learning. There are many notable exceptions, amongst them the techniques employed in CCNs.

A rough outline of the CCN procedure is as follows:

1. Begin with a single layer neural network. Repeat until a measure of convergence is achieved:
    a. Train existing output layer connection weights (parameter learning) to reduce error on training data (supervised learning)
    b. Insert a new hidden layer node, with inputs from all existing inputs and hidden layer nodes

in the network (cascaded). Note that the output of this node is not yet connected to the output layer nodes of the network.

c. Train the input weights of this new node to maximize the absolute correlation of the node's output to the error of the existing network's output on the training cases (supervised learning).

d. Connect the output of the new node to all output layer nodes.

At first, this may not seem to meet the discrete-optimization criteria for structural learning discussed above. However, note that input layer node weights are often adjusted to values near zero in step c above. In effect, this "turns off" the connection between a hidden layer node and a particular input (or another hidden layer node). Moreover, there are variations of CCN that employ a population of randomly-initialized nodes in step b and c, culling this population to one node in step d. If these nodes are initialized with less-than-full input connectivity, the discrete optimization of structural learning is clear.

CCN is relatively straightforward algorithm, with the exception of the somewhat mysterious element of step c. One must consider why one would *maximize* the absolute correlation between a node's output and existing network error. Upon consideration, it becomes clear that this step will allow one to cancel out that existing error with the new node, through the weight adjustments in step a.

2.3 CCN and XCS

XCS, perhaps the most popular LCS paradigm, has a notable similarity to CCN. One of XCS's innovations is its use of accuracy, a second order statistic, as fitness. Similarly, CCN attempts to maximize absolute correlation, another second order statistic, in its creation of hidden layer nodes. If one imagines an analogy where XCS rules are like hidden layer nodes [13], there is a clear correspondence.

However, it is not an exact correspondence, leading one to ask why a difference exists. Note that in CCN, correlation to existing error is justified by supervised learning training of the output layer weights, to cancel out that error. No such "cancellation" exists in XCS, since a rule's "output" (action) is not tuned via supervised learning.

2.4 Supervised Versus Reinforcement Learning

XCS grows out of the LCS tradition of reinforcement learning [14]. Reinforcement learning is defined by the lack of supervisory feedback that indicates the correct action the system should take. Instead, only "reward" or "punishment" type feedback is available. Since XCS grew from reinforcement learning, it generally does not employ supervised update of actions. Instead, actions are searched for via the GA, or various covering operations.

However, XCS has been applied to many supervised learning problems. In supervised learning problems direct feedback is available that indicates the correct actions for the system (often via solved training cases). CCN exploits this supervision in its update of output layer weights, which justifies its use of correlation to existing error in hidden layer weights.

This suggests the analogy upon which MILCS is built. However, rather than employing correlation, we have employed mutual information (yet another second order statistic). We feel this provides an additional theoretical backing for the system, which is discussed below.

3. The Role of Mutual Information in MILCS

Shannon's fundamental work in information theory [11][12] addresses the following concern: given an input signal to a communication channel, how does one maximize the rate of communication, while minimizing error in that communication? This is closely related to Shannon's work on data compressing, which considers the maximum compression ratio for lossless representation of data. Both problems derive similar results.

Shannon showed that the zero-error maximum communication rate for a channel is given by maximizing the mutual information between the channel's input and output. Maximization of mutual information is accomplished by manipulation of the probabilities of various inputs to the channel, or through the manipulation of the coding of input signals. Since coding is similar to compression, the analogy to lossless compression is clear.

Imagine that the existing error in step c of the CCN procedure is an input signal to a communication channel. In this case, the hidden layer node plays the role of an encoder for signal. Therefore, we find a firm theoretical foundation for using the mutual information as the fitness of this node, through Shannon's theorems.

Another useful analogy is to sensor placement. Imagine that one is set the task of placing temperature sensors in a large space, with the goal of delivering maximum information about the temperature distribution in that space. If one can estimate the probability distribution of temperatures over the space, and one knows the response field of the sensors, one can maximize mutual information between these distributions to optimally place the sensors. This is similar to the placement of the conditions of classifiers (or the receptive fields of neurons).

Mutual information between variables X and Y is given by:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = \sum_{y \in Y} \sum_{x \in X} p(x \mid y) p(y) \log \frac{p(x \mid y)}{p(x)} \quad (0.1)$$

It is useful to examine the terms of this expression. Consider x to be the event of an error, and y to be the event of a particular rule matching. Thus, $p(x)$ is the distribution of existing error of a system, and $p(y)$ is the distribution of responses (matching or not matching) of a rule. In this case, the first term, $p(x \mid y)$, can be seen as the relevance of the rule's output to existing error. The second term, $p(y)$, is the generality of the rule. The third term, $\log(p(x \mid y)/p(x))$, is a comparison of the error when the rule matches to the overall error in the space. This is a sort of specificity term. Thus, the mutual information expression offers a

balance of accuracy, generality, and specificity, in an optimal fashion dictated by Shannon's Theorems.

4. The MILCS Process

Given the above considerations, MILCS operates as follows. Note that to conform to the CCN analogy, and articulate all the terms in the sums of equation 1.1, we have given each rule two actions: one for when the rule matches, and one for when the rule does not match. Both are updated via simple supervised learning:

Starting with a set of random rules

1) For each random training case, as follows:
   a) Repeat the following for all rules:
      i) Remove a rule from the population set
      ii) Do action selection based on prediction values of all 'mature' rules (rules which have been trained for more than a threshold number of training cases, excluding the removed rule) of both matched rule set and not-matched rule set and calculate reward based on the chosen action
      iii) Update counters necessary for modeling an empirical probability distribution over the matched/not-matched condition of the removed rule, and the error of the remaining rules. MI is calculated from this probability bivariate distribution.
      iv) Add that rule back to the population set
   b) Do action selection based on prediction values of all 'mature' rules (rules which have been trained for more than a threshold number of training cases) of both matched rule set and not-matched rule set (action with the highest prediction wins) and calculate reward based on the chosen action
2) Update actions (output) and prediction values of all rules based on the reward and their previously selected actions (supervised learning). Note that in the current experiments, we simply keep track of which action yields a higher reward (for both the matched and not-matched condition), and set the rule's action to that value. More sophisticated supervised learning could be used in this step.
3) Calculate and update the fitness of the action set.
4) Determine and flag all the rules of the population set that have been trained above another threshold, which determines whether rules are 'mature' enough to subsume other rules and participate in a non-panmictic GA.
5) Subsume rules in the action rule set to keep the population compact. Rule A subsumes rule B if:
   a) Both rules are above the maturity threshold for subsumption,
   b) Both rules have the same action (determined by supervised learning),
   c) A is more general than B,
   d) A has a reward prediction greater than or equal to B. This step varies with testing problems
6) Perform a non-panmictic GA: Select based on these mutual-information-based fitness values and subsume the offspring to the parents if possible (using the subsumption conditions previously outlined). Children rules are added to the rule set. If the rule set size exceeds a maximum, rules are deleted based on the amount of time since their last inclusion in the action set. To be deleted, rules must have participated in action selection more than "deletion activation threshold" number of trials, and they must not have been in the highest-prediction rule for the last "deletion threshold" trials.
7) Reset the MI counters of action rule set.
8) Repeat from 1), until some convergence criteria is met.

Note that the removing and adding procedures (step 1)a)) are there to conform of the CCN analogy that the new hidden layer node is not yet fully connected to the network thus the new rule's future parent should not have any effect on the system

5. Results

We have tested MILCS on the multiplexer problem and on the coordination number (CN) protein structure prediction problem.

5.1 Multiplexer Problems

We have evaluated results on the 6, 11, and 20 multiplexer problems. In order to show a thorough comparison, we present these results, along with results obtained from XCS (using [4]). Three lines appear on each plot: the percentage correct over the past 50 training cases (solid line), the difference between reward and predicted reward over the past 50 training cases (dashed line), and the number of macro-classifiers (unique classifiers) in the population (dash-dotted line) divided by 1000. Graphs reflect the average of 10 runs.

For XCS, we employ the parameter settings reported in [16]. MILCS parameter settings are shown in Table 1.

Table 1: MILCS parameters for the multiplexer problems

Multiplexer problem size	**6**	**11**	**20**
Maximum Pop Size	50	510	500
GA Maturity Threshold	5	25	125
Probability of #s	0.33	0.33	0.66
Initial Pop Size	30	50	50
Action Selection Maturity Threshold	25	1024	10500
Maturity for Subsumption	16	330	2100

Figure 1 and Figure 2 show results from XCS and MILCS applied to the 6 multiplexer. Note that MILCS converges more rapidly, and to a smaller final population of unique classifiers.

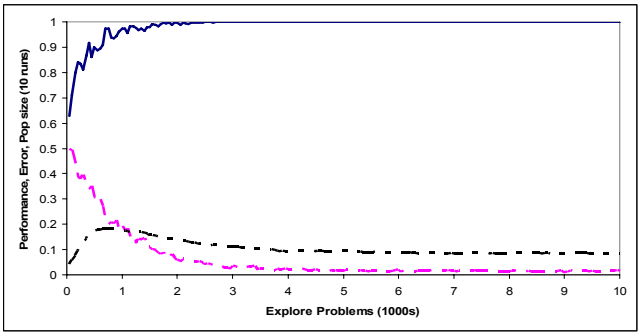

Figure 1: Results from XCS applied to the 6 multiplexer

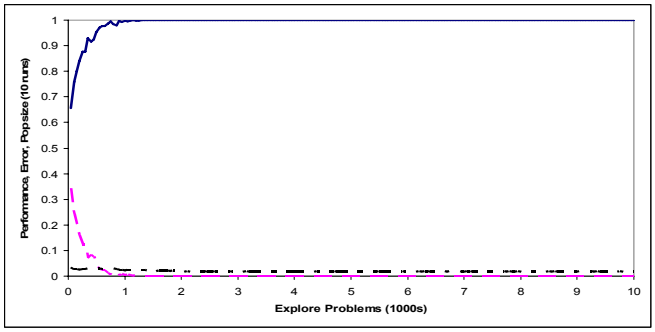

Figure 2: Results from MILCS on the 6 multiplexer

Figure 3 and Figure 4 show results from XCS and MILCS applied to the 11 multiplexer. While convergence times are similar, MILCS still converges to a smaller final population of unique classifiers.

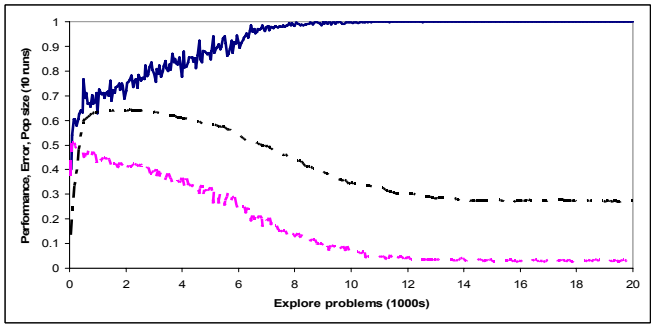

Figure 3: Results from XCS on the 11 multiplexer

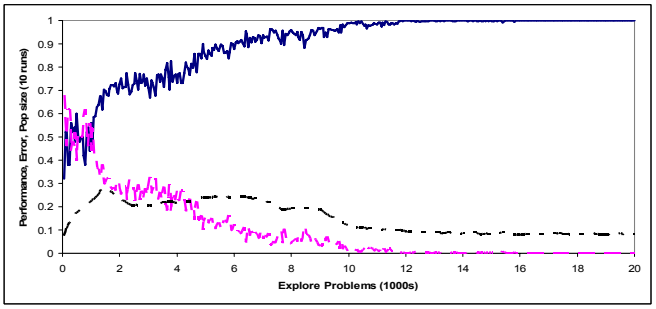

Figure 4: Results from MILCS on the 11 multiplexer

Figure 5 and Figure 6 show results from XCS and MILCS applied to the 20 multiplexer. In this case, XCS converges more rapidly, but MILCS maintains a smaller final population of unique classifiers as before.

However, we note that with XCS and MILCS, we ran complete tests on all possible inputs for each of the multiplexer problems. Each system passed this "full test" in each situation, with the exception of XCS applied to the 20 multiplexer, which failed on a small number of cases at the end of some runs portrayed in the average of 10 shown in Figure 5. While we did not overcome this difficult with the code provided in [2], we were able to reproduce perfect behavior in approximately 75,000 explore problems using [10]. This is consistent with the results on XCS scaling for the multiplexer problems provided in [18].

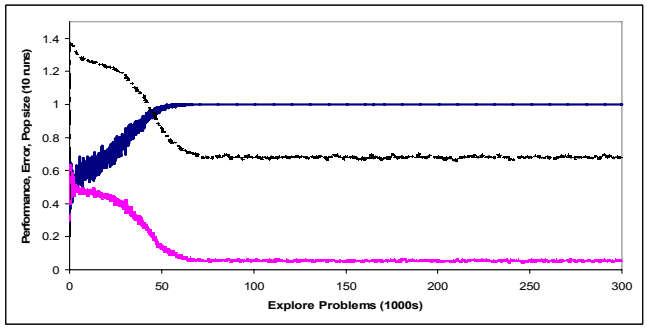

Figure 5: Results from XCS on the 20 multiplexer.

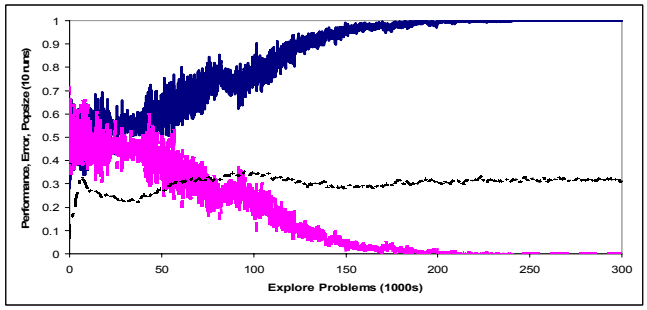

Figure 6: Results from MILCS on the 20 multiplexer.

6. Scalability

Our results reveal MILCS scaling up slightly worse than XCS (see Figure 7)

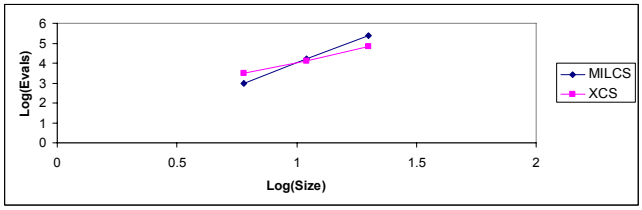

Figure 7: Log-log plot of polynomial scaling of MILCS and XCS on the multiplexer problems.

However, both are scaling as low-order polynomials. While there is an apparent difference in the scalability between the algorithms, it is not of significant order. Moreover, the previous results indicate that MILCS is ending with a substantially smaller set of unique classifiers. Hand examination of rule sets has revealed that almost all of the mature rules at the end of our MILCS runs are best-possible-generalized rules for the multiplexers. However, this examination relies on our knowledge of the underlying problem. In the following section, we introduce a method of visualizing the comparative explanatory power of the final rule sets.

7. Explanatory Power

While accuracy and compute time are important metrics for machine learning systems, it is also important to consider their *explanatory power*. While this term has no formal definition, it is considered to be the subjective human understandability of the representation of the knowledge learned by the system. As a subjective quality, it is somewhat difficult to present, particularly when the knowledge representation exists in a highly multi-dimensional space. It is also important that we avoid using out pre-

existing knowledge of a problem's structure in evaluating the explanatory power of resulting knowledge representations.

7.1 Visualization of Explanatory Power

In an attempt to visualize the relative explanatory power of knowledge representations, we have developed the following procedure. Consider a structural element of the knowledge representation (in particular, a rule). We will represent this rule as a circle, where the diameter reflects the element's generality. We will characterize the overlap of the receptive fields (conditions) of these elements by the overlap of the circles. The color of the circles will represent their output (actions). If rules do not overlap, we consider the difference between them to determine the relaxed spring distance between corresponding circles.

Each circle will act as a simulated mass, connected to the other circles via springs, whose spring force is zero when the desired overlap is obtained. Dampers between the circles are added (to insure convergence). Given this, we can use a simple dynamic systems simulation that iterates to an equilibrium, which will represent the nature of the knowledge in the system in a simple, two-dimensional space.

To make a valid comparison that is not built on pre-existing knowledge of the problem at hand, the visualization will include all elements (rules) that play a role in the output determination of the final system. In XCS exploitation mode, the following factor is computed:

$$\sum(\text{prediction} \times \text{fitness}) / \sum(\text{fitness})$$

over all matching rules, for all actions, and the action with the highest factor is selected. Therefore, since all the rules participate in action selection, we include all these rules in our visualization.

However, in MILCS on an "exploitation" trial, only the rules with action-selection maturity above a threshold are employed. The rule, either matched or not matched, with the maximum predicted reward is always selected as the rule that acts. Therefore, only these rules are used in our visualization.

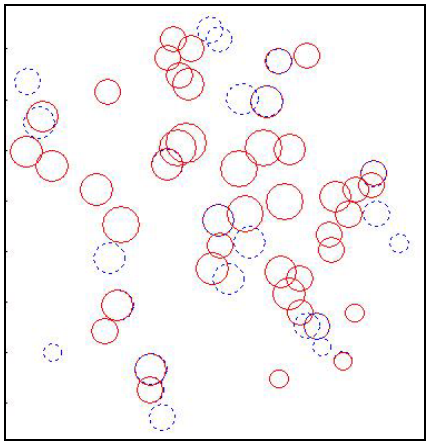

Figure 8: Visualization of the final rule set developed by XCS on the 6 multiplexer.

Figure 8 and Figure 9 show visualizations of the final rule sets from XCS and MILCS (respectively) applied to the 6 multiplexer. The smaller, final MILCS rule set, and its inclusion of only perfect generalizations is clear. We believe that this visualization shows the superior explanatory power of the resulting rule set in a way that does not depend on human understanding of the rule-form of a correct final solution for this particular problem.

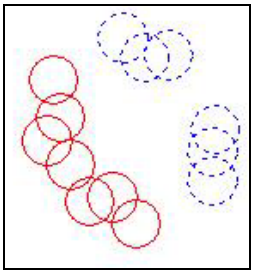

Figure 9: Visualization of the final rule set developed by MILCS on the 6 multiplexer.

Figure 10 and Figure 11 show visualizations of the final rule sets from XCS and MILCS (respectively) applied to the 11 multiplexer. Once again, the superior explanatory power of the MILCS rule set is apparent. As in the 6 multiplexer, a decision surface between the two actions is apparent, even after the projection of the rule set to a two dimensional space.

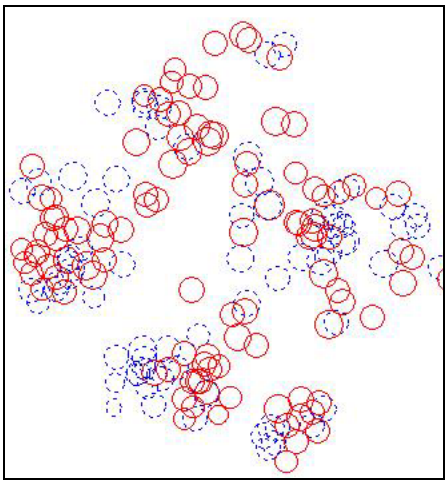

Figure 10: Visualization of the final rule set developed by XCS on the 11 multiplexer.

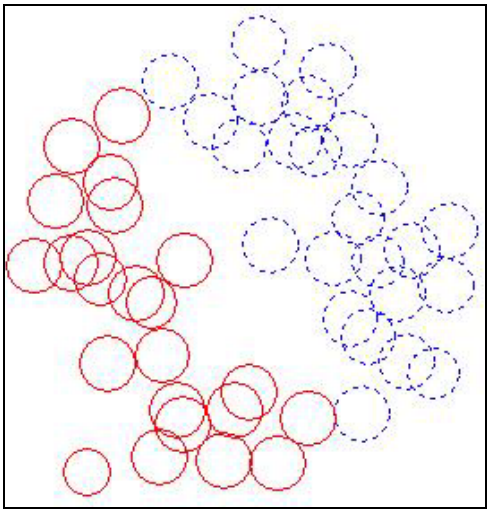

Figure 11: Visualization of the final rule set developed by MILCS on the 11 multiplexer.

Figure 12 and Figure 13 show visualizations of the final rule sets from XCS and MILCS (respectively) applied to the 20 multiplexer. While a linear decision surface is no longer apparent in the MILCS result, the less complex structure of the MILCS rule set when compared to the XCS rule set is apparent. Note that while only some small portion of the final XCS rule set are perfect generalizations, and a human could detect these with knowledge of the multiplexer. That would not be the case in a problem of unknown structure.

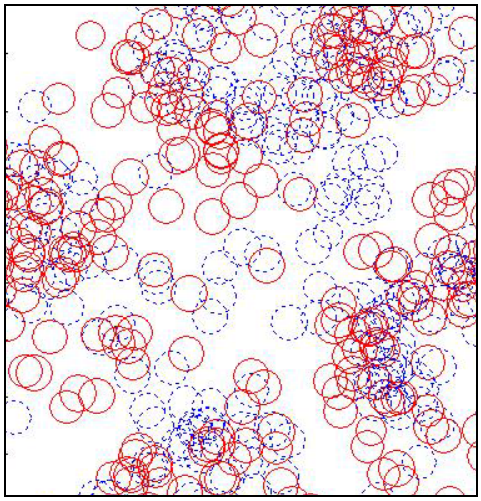

Figure 12: Visualization of the final rule set developed by XCS on the 20 multiplexer.

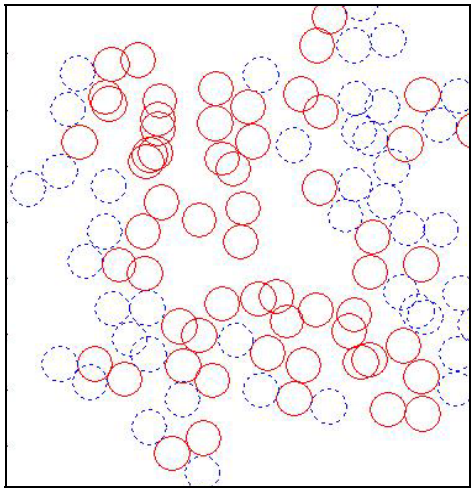

Figure 13: Visualization of the final rule set developed by MICS on the 20 multiplexer.

7.2 Coordination Number Problem

Coordination Number Prediction is one of the popular approaches to prediction the 3D structure of a protein. It is defined as the prediction, for a given residue, of the number of residues from the same protein that are in contact with it. Two residues are said to be in contact when the distance between the two is below a certain threshold. A simplified protein model, the HP model, has been used to understand protein structure prediction. This model represents the sequence of a protein using two residue types: hydrophobic can polar. We have produced preliminary results on this simplified, two-state CN prediction of real proteins with source data provided by Stout et al. in [15]. In order to show a thorough comparison, we present these results, along with results obtained from XCS (using [2]) and GAssist [1]. GAssist is a Pittsburgh-style learning classifier system which has been recently used for many protein structure prediction problems and achieved significant results [15]. Accuracy results presented here use the same training and testing procedures as those in [15].

We tested all three classifier systems on 3 different window sizes for 2 state predictions.

For XCS, we employ the same parameter settings for the 20 multiplexer problem whereas MILCS parameter settings are shown in Table 2.

Table 2: MILCS parameters for the CN problem

Window size	1	2	3
Maximum Pop Size	100	200	300
GA Maturity Threshold	25	25	25
Probability of #s	0.66	0.66	0.66
Initial Pop Size	50	50	50
Action Selection Maturity Threshold	450	450	450
Maturity for Subsumption	400	400	400
Deletion Threshold	1600	1600	1600
Deletion Activation Threshold	450	450	450

GAssist results were generously provided by the authors of [15]. Results are shown in Table 3. In XCS and MILCS, the representation was one bit each for the "H", "P", and "not present" conditions of the neighboring residues, and one bit each for the "H" and "P" conditions of the target residue. For instance, with window size 1 this yields a 7-bit condition, given that there are two neighboring residues (3 bits each), and the target residue (2 bits). This sparse encoding was used for consistency with the GAssist representation.

Table 3: Results of XCS, MILCS, and GAssist on two-state (HP) CN problems of various window sizes.

Window Size	Method	Accuracy	Final Rule Size	Max Evals
1	XCS	60.3% ±4.7%	53.1	100000
1	MILCS	63.5% ±0.5%	8.7	100000
1	GAssist	63.6% ±0.6%	4	8000000
2	XCS	61.1% ±3.6%	197.8	100000
2	MILCS	63.8% ±0.6%	20.6	100000
2	GAssist	63.9% ±0.6%	4.4	8000000
3	XCS	61.6% ±3.3%	371.2	250000
3	MILCS	63.0% ±0.8%	55.2	100000
3	GAssist	64.4% ±0.5%	4.8	8000000

Note that for each algorithm, "Max Evals" is the number of evaluations in a run, and this most likely represents a high upper bound on the number of evaluations required to get results of the

indicated quality. However, these numbers do indicate the relative order of magnitudes of convergence times for results shown. Rule set size reflects the number of rules participating in action selection at the end of each run. MILCS performance is statistically similar to GAssist, which provides the best results. MILCS results in larger rule sets that GAssist, but converges an order of magnitude faster. XCS does not perform as well as MILCS or GAssist overall. While these results are preliminary, and we feel we can improve MILCS performance, they are promising.

Final Comments and Future Directions

Preliminary results with MILCS are promising, with respect to accuracy, speed, and explanatory power. While MILCS seems to scale slightly worse than XCS, this may not be an entirely fair comparison, since our preliminary results show that MILCS finds a smaller, more explanatory rule set. We believe this superior effect is to be expected, given the firm information theoretic basis of the mutual information fitness function.

Evaluating the system on more problems is the clearest direction for further investigation.

However, the concepts in MILCS are not specific to the particulars of a rule learning system. Exploring a neural network system that employs a similar structural learning paradigm is also a promising direction for future investigation. The use of mutual information in this fashion may also have application in supervised learning of other knowledge representations.

While the focus of this work has been on supervised learning, it is possible that the system may be adapted to reinforcement learning. Note that to some extent, XCS already adapts reinforcement learning to supervised learning, in its tendency to learn a complete "model" of the long term payoff function across the state/action space. The mapping from state/action to payoff is a supervised learning problem, drawing on Bellman optimality and Q-learning the appropriate target values and error functions.

It will also be interesting to further investigate the visualization technique employed in this paper to compare explanatory power in a larger variety of knowledge representations.

8. ACKNOWLEDGMENTS

The authors gratefully acknowledge support provided by the Engineering and Physical Sciences Research Council (EPSRC) under grant number GR/T07541.

9. REFERENCES

[1] Bacardit, J. *Pittsburgh Genetics-Based Machine Learning in the Data Mining era: representations, generalization, and run-time* (2004). PhD thesis, Ramon Llull University, Barcelona, Catalonia, Spain.

[2] Bernadó-Mansilla, E (2003). Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. Evolutionary Computation 11(3).

[3] Butz, M. V.(203) Documentation of XCS+TS C-Code 1.2. *IlliGAL report 2003023*, University of Illinois at Urbana-Champaign. (Source code: ftp://gal2.ge.uiuc.edu/pub/src/XCS/XCS1.2.tar.Z)

[4] (2003) ftp://gal2.ge.uiuc.edu/pub/src/XCS/XCS1.2.tar.ZXCS (+ tournament selection) classifier system implementation in C, version 1.2 (ftp://gal2.ge.uiuc.edu/pub/src/XCS/XCS1.2.tar.Z) (for IlliGAL Report 2003023, University of Illinois Genetic Algorithms Laboratory).

[5] Fahlman, S. E., and Lebiere, C. (1990). The Cascade-Correlation learning algorithm. In *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann.

[6] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

[7] Heckerman, H., (1996). A Tutorial on Learning with Bayesian Networks, Technical Report, MSR-TR-95-06.

[8] Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems* 2nd Edition. MIT Press, Cambridge.

[9] Holland, J., Holyoak, K. J., Nisbett, R. E., and Thagard, P. 1986. *Induction: Processes of Inference Learning and Discovery. Cambridge*. MIT Press.

[10] Lanzi, P. L. *xcslib:* source code available at http://xcslib.sourceforge.net/.

[11] Shannon, C.E. (1948). A Mathematical Theory of Communication, *Bell System Technical Journal*, 27, pp. 379–423 & 623–656, July & October, 1948

[12] Shannon, C.E. (1949). Communication in the presence of noise, Proc. Institute of Radio Engineers, vol. 37, no.1, pp. 10-21, Jan. 1949.

[13] Smith, R. E. and Cribbs, H. B. (1994). Is a classifier system a type of neural network? *Evolutionary Computation*, 2(1), 19-36.

[14] Sutton, R. S. and Barto, A. G. (1998) *Reinforcement Learning: An Introduction*. MIT Press.

[15] Stout, M., Bacardit, J., Hirst, J., Krasogor, N. and Blazewicz, J. (2006). From HP lattice models to real proteins: coordination number prediction using learning classifier systems. In *4th European Workshop on Evolutionary Computation and Machine Learning in Bioinformatics 2006*.

[16] Wilson, S. W. (1994). Classifier Fitness based on Accuracy, *Evolutionary Computation 3(2)*. pp 149-175.

[17] Wilson, S. W. (1994). ZCS: A Zeroth-Level Classifier System, *Evolutionary Computation 2(1)*. pp 1-18.

[18] Wilson, S. W. (1998) Generalization in the XCS classifier system. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., and Riolo, R. (eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann