Collective Behavior based Hierarchical XCS

Matthew Gershoff
Matthew Gershoff Consulting
New York, New York
mattg@mattgershoff.com

Sonia Schulenburg
Level E Limited and Centre of Intelligent Systems and their Applications
Edinburgh Technology Transfer Centre,
The King's Buildings, Mayfield Road,
Edinburgh, EH93JL, United Kingdom
sonia@levelelimited.com

ABSTRACT

This paper attempts to extend the XCS research by analyzing the impact of information exchange between XCS agents on classifier performance. Two types of information are exchanged and combined to improve classification performance. The first uncovers information contained in the signal patterns of collections of Homogeneous XCS classifiers. This information is used to determine which subsets of the state-space the XCS can be expected to be accurately classified. The second combines the results of XCS agents that are each tasked to solve different portions of the original problem. Results on the multiplexer (6, 11) indicate that given accurate problem domain assumptions, the Collective Behavior (CB-HXCS) method shows promise. Results show - at least in simulated multiplexer environments - that the HXCS is able to solve a well defined problem with less data than an individual XCS. This approach seems very promissing in real-world applications where data is incomplete, expensive or unreliable such as in financial or medical domains.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning - concept learning, knowledge adquisition.

General Terms

Algorithms, Experimentation.

Keywords

Multiplexer, Information exchange, XCS, CB-HXCS, Voting, Classifier, Collective Behavior.

1. INTRODUCTION

The objective of this paper is to extend XCS research by analyzing the impact of information exchange on classifier performance. It is proposed that by analyzing the collective behavior of the classifiers it is possible to improve upon performance over individual classifiers. The two main collective learning questions to be addressed are:

1. Is it possible, based only on their collective signal pattern, for a set of homogeneous XCS agents to determine which portions of a problem space they are best able to solve?

2. Can global XCS performance be improved by solving and combining smaller sub-problems?

The idea behind XCS is that the end product of combining accuracy and a niche GA results in a complete and accurate mapping of X x A => P from inputs and actions to payoff predictions. XCS evolves maximally general classifiers subject to an accuracy criterion, so that the mapping gains representational efficiency [5]. Due to space limitations description of XCS has been omitted. Refer to [6] for a full review.

The classic paper entitled "A Critical Review of Classifier Systems" [7] gives a good summary of the unsolved problems and new challenges that LCS faced in the late 80s. Since then, there have been great accomplishments in theoretical aspects (mapping performance and generalization), of XCS solving a variety of single-step environments such as the Boolean multiplexer and sequential environments (multi-step) like the woods-type of problems.

In many problem domains, including the multiplexer, the maximum number of dimensions in which an optimal rule sits is often less than the dimensions of the state-space. In the case of the multiplexer-6 problem the rules are sitting in three dimensional subspaces across all six dimensions of the state-space. If the state-space was partitioned in smaller subspaces equal to the size of the maximum rule length, then it might be possible to find each of the rules sitting in each partition.

To solve the global problem, the results of each of the lower level problems need to be combined. This can be done by creating a hierarchy of XCS agents where agents at the next level learn from the signals emitted from the agents at the lower level. However, since the low-level agents are only able to solve a portion of the environmental states, there needs to be some mechanism for the agents to indicate to the higher level agent when to be confident of the signal accuracy. In order not to bias the result, this confidence measure should be made at each step without the use of the true classification.

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

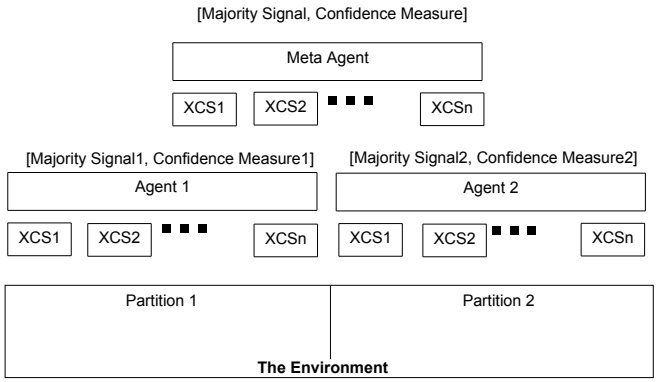

Figure 1: CB-HXCS Framework

2. THE MODEL

The CB-HXCS process is shown graphically in figure 1. The environment is first partitioned in to smaller subspaces. A base level agent is then assigned to each data partition. Each agent is comprised of a fixed set of XCS micro-agents. These micro agents are passed the same environmental state at each time step. Each micro agent emits a classification (action) signal. The signals are each treated as a vote for a specific classification. The signal that receives the majority (or plurality, for when there are more than two options) of votes is designated as the aggregate signal for the agent.

2.1 Voting

This approach is based on the assumption that the best expectation of an agent's future result is the average value over all possible runs of the agent. The majority rule is a simple voting method that enables the collection of homogeneous agents to present a single classification signal. Not only does the use of a set of micro agents enable us to average the results of the XCS, but it also enables creation of a measure of how certain the group is of its vote.

2.2 Confidence

Along with the majority signal, the aggregate agent is able to emit information about the competitiveness of the vote. If it is assumed that movement away from the average is because of stochastic events, then we may conclude that the greater the vote differential, the greater the confidence in the winning signal. For example, given an environmental state, the likelihood that 100 micro-agents would all signal '1' given that the true probabilities are equal (0.5), should be quite low. Of course this does not mean that a larger vote differential implies a class distribution that is radically different from uniform. It might be the case that even with a distribution of 0.55 and 0.45, all agents would select the class that occurred with a probability of 0.55.

In [4], the author argues that successful ensemble classifiers require that the component classifiers have error rates below 0.5 and be uncorrelated with one another. This, however, assumes that the classification rate is consistent over the underlying input data's distribution. If an agent is able to gain a reflective capability about its own accuracy rate through the level of classification consensus, then it may be able to signal to other agents when it believes that is able to classify a case well.

Such an indicator can be constructed by categorizing signals that win with a vote margin over a predetermined threshold. Higher voter margins of victory should be less likely when the conditional probability of a class, based on the environmental state, is the same as naive class probability. When the environmental subspace is solvable, then the margin of victory should tend to be very large.

2.3 Combining Heterogeneous Agents

The aggregation of signals from homogeneous micro agents to reduce classification variance is one way in which agents can communicate across the social group. It is possible to envisage situations where it would be advantageous for each of the aggregate agents to pool their learning to improve global classification. One can then construct a network of meta agents (see figure 1) that combine the signals from the lower level agents into a final classification signal. The approach is a form of ensemble learning. Work by [4], [8], and [1] has looked into different methods (bagging, boosting, and stacking) of combining classifiers to produce a final meta classifier. As with [3], the method employed here uses an on line learning procedure, the XCS. This eliminates the need for the various re-sampling methods needed for Bagging and the hold-out samples needed for Boosting - (Boosting constructs the set of classifiers in a step-wise fashion, the classifiers are dependent on the results of the previous classifiers, a potential over-fitting problem).

The meta agent is displayed at the top of figure 1. The Meta Agent treats the set of majority signals and confidence indicators as its search environment. As with the base level agents, the meta agent is comprised of a collection of micro agents - the only difference between the two types is that the meta agent only searches over the output of the base agents rather than the original environment. The complete process CB-HXCS process is

1. Partition the environment space (independent variables) into smaller subspaces.
2. Expose the micro agents to the initial environmental state from the appropriate partition
3. Collect each micro agent's classification signal for the given state.
4. Calculate the winning vote and the margin of victory for each set collection of micro agents
5. Expose the majority signal and confidence indicator to each appropriate meta agent collection.
6. Emit the final classification signal from the top level meta agent.

The XCS agents (both the micro and meta agents) all receive the appropriate reward value for their action only after every agent in the network has emitted its classification signal. This ensures that information about future rewards does not leak into the system, which could lead to over-fitting. The entire chain can be run in an on-line environment without the need for out-of-sample testing.

One question that arises from the above approach is that even if the CB-HXCS is able to perform at the same level as the Single XCS on the full data set, why bother? The answer is that for some problem domains the CB-HXCS may

be able to solve the problem with less data than the full XCS approach and at the same time lends itself to a more distributed computing model as presented in [3]. For problem domains where the marginal cost of data is high relative to computation time then the CB-HXCS approach may be useful.

In order to compare the learning rates and overall accuracy of the CB-HXCS to the XCS with full data access, 6 and 11 bit multiplex problems were both used. In addition, we will compare the result of the CB-HXCS to the aggregated agent results for each of the data partitions. Superior performance from each of the higher level agents would indicate that it is possible for agents to share information via emission and confidence signals. Finally, a network without the confidence measure is evaluated in order to determine if the confidence measure improves the network's performance.

3. IMPLEMENTATION

The implementation of the XCS was based primarily on Butz's XCS implementation in Java, version 1.0. This code was developed at the University of Illinois Genetic Algorithm Lab [2]. While this code provides basic XCS functionality, minor modifications and extensions were needed for the CB-HXCS method.

1. The environmental data is passed sequentially to the XCS, rather than randomly selecting the state. This ensured that each XCS in the network was evaluating the same environmental state with respect to the other XCS processes in the group

2. The system was also altered so that even when the XCS was exploring, the signal passed along to the voting process was always the exploit signal. The XCS still runs the normal explore method for its own internal process

3. Unlike the standard XCS, the CB-HXCS is sensitive to the explore/exploit rate. The results use a decreasing explore rate ($\text{rate}_t = .999 * rate_{t-1}$) - unless otherwise noted

4. A threshold parameter used to set the confidence signal has also been added. Values range between 0% and 100%, with higher values indicating greater vote margins. For simplicity, this was set at a fixed 80% for all of the agents in each model

5. The standard XCS parameters were all set to the default levels as per Butz's implementation - with the number of micro-classifiers set to 800

A common test environment for the XCS is the multiplexer. The multiplexer is a problem in which the first k bits encode which of the remaining n-k bits contain the solution. For example, for a multiplexer-6 the bit string '001101' would have a solution of '1' as the two left-most bits '00' represent the first, or right most bit of the string.

The multiplexer problem is one that can be separated into smaller sub problems. The multiplexer-6 can be thought of as four sub-problems of length three and the multiplexer-11 can be broken down into eight sub problems of length four. Of course the partitioning of the original environment into sub problems is in effect a bias -or information- that the researcher introduces into the solution. For real-world problems the determination of the maximal rule length and the decision of how to partition will often be based on assumptions about the problem domain, and may not be correct. Assumptions about the reduced complexity of a problem domain is roughly analogous to the selection of fractional factorial designs, where the interactions of only a subset of variables are considered.

The multiplexer-6 problem was modeled with a 4-2-1 agent hierarchy. The four base level agents pair up and report to one of two second level meta agents. These meta agents in turn report to the final agent. Based upon knowledge of the problem, the data was partitioned into four subsets. Each partition included the first two bits plus one of the remaining four bits. For example, partition1 would cover bits one through three, partion2 bits one, two, and four, etc. The multiplexer-11 problem was set up analogously. The data was partitioned into eight subsets with an 8-4-2-1 agent hierarchy.

Each of the first level meta agents receives the winning classification and confidence signal from two of the low level base agents. The meta agents therefore cover a four dimensional state-space. The classification and confidence signal of the two pairs of first level meta agents are in turn presented to the two second level meta agent and so on. The final agent presents the final classification.

For the multiplexer problem it is possible to know a priory how to partition the data. In many cases this will not be so clear. If only the maximal rule length is assumed, and there is no other information to partition the data, then the number of base level agents needed to span all possible rules of that length grows to n!/(r!*(n-r)!), where r is the maximal rule length and n is the number of original dimensions. For the multiplexer-6 problem, given a maximum rule length of 3, then the number of base level agents needed to ensure covering the all potential rules is 20.

In order to cover all possible rules of length four (without making any additional assumptions about partitioning) in the multiplexer-11 problem, there needs to be 11!/(4!*(11-4)!)(330) base level agents. As the size of the environment increases – and the closer the maximal rule length is to n/2 – the number of agents needed to cover all potential rules increases.

One possible trick to reduce the agent quantity is to slightly increase the size of each partition. For example, a six dimensional partition includes 15,(6!/(4!*(6-4)!), four dimensional sub spaces. Because each of the four dimensional subspace will sit in more than one of the six dimensional partitions, the 330 agents in the multiplexer-11 problem can not be whittled down to just 22. To have at least one example of each four dimensional combination, a minimum of 40 base agents is required. To test if this approach has merit, a 40-8-2-1 model is constructed and run along with the other tests.

4. RESULTS

The expectation that the voting signal patterns should be different for 'learnable' and 'unlearnable' regions of the search space appears to in fact be the case. The voting patterns of the base level micro agents have been combined and shown in figures 2 and 3. Figure 2 displays the frequency of cases by vote percentage of classifying a case as a '1' in the unlearnable regions of the space. Notice how the pattern

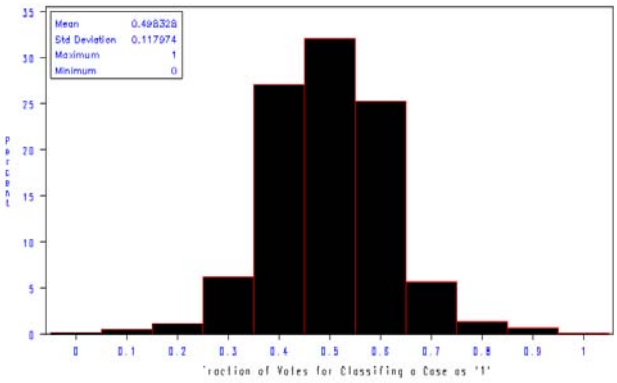

Figure 2: Histogram of Voting Pattern in 'Unlearnable' Regions

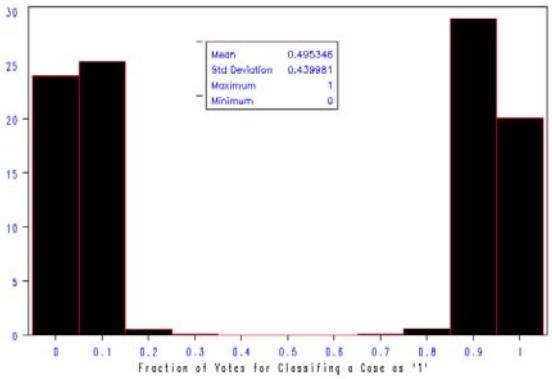

Figure 3: Histogram of Voting Pattern in 'Learnable' Regions

tends to follow a normal distribution with the center at 50% and with few cases out in the tails. This stands as a sharp contrast to the voting patterns in the 'learnable' regions as seen in figure 3. Almost all of the cases are sitting out in the tails with margins of victory at or above 80%. These results suggest that cases with a classification consensus will be more likely to fall into a region of space that the agent is able to accurately classify.

Figure 4 displays the results for all 15 agents used by the CB-HXCS in the multiplexer-11 problem. The graphic shows that the agents of each higher level area are able to learn from and outperform the agents under them in the hierarchy.

The dotted lines at the bottom of the chart display the accuracy rates of the base level agents. The expected accuracy for these agents is 56.25%. This is because approximately 12.5% of the cases should always be accurately classified by each of the agents, with the remaining 87.5% correctly assigned 50% of the time. The four series above the base agents are the results of the first level meta agents. Each meta agent using only data emitted from two of the base

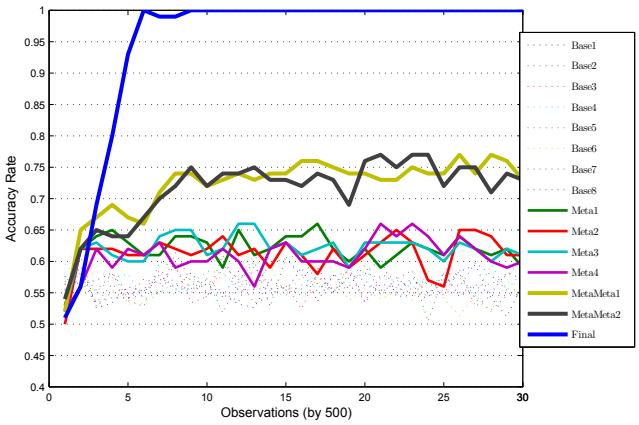

Figure 4: Multiplexer-11: CB-HXCS All Agents

level agents, is able to improve the accuracy results to approximately 62%. The two agents at the penultimate level are able to achieve close to 75% accuracy. The agent at the top of the hierarchy reaches 99% accuracy after only 2,500 observations and 100% accuracy after 4,000 observations.

To see how the CB-HXCS compares to the single XCS approach, The multiplexer 6 and 11 problems are run with both methods. The CB-HXCS is able to solve the multiplexer-6 problem after around 2,500 observations (see figure 5). The Full XCS model reaches 100% accuracy after only 300 observations. It is not surprising that the simple XCS outperforms the more complicated CB-HXCS model in small problem domains.

To determine if the confidence measure improved performance, a model was included that only emitted the majority signal to the agents in the next level of the network. The performance of this model on the multiplexer-6 problem was quite poor. The average accuracy for the final 5,000 observations was only 64% with a standard deviation of just over 5%. This result supports the hypothesis that the inclusion of the confidence signal significantly improves the overall results of the network.

In the larger multiplexer-11 problem the CB-HXCS is able to outperform the full XCS model. Both the CB-HXCS and the full XCS are shown in figure 6. The single XCS reaches full accuracy at 9,000 observations compared to only 4,000 for the CB-HXCS model.

In addition to the models with expert guided partitions, two additional models were constructed that make no assumptions of how to subset the data -the full combinatorial and the reduced combinatorial partitions. The results of the full combinatorial CB-HXCS model in the multiplexer-6 problem are shown in figure 5. Given only that the length of the largest needed rule is three, the full combinatorial CB-HXCS is able to solve the multiplexer-6 problem using only slightly more data observations then the expert partitioned CB-HXCS model.

The reduced combinatorial partitioning was applied to the multiplexer-11 problem. The results, shown in Figure 6, are mixed. The model is unable to consistently solve the multiplexer with 100% accuracy, even though it does on occasion reach 100%. It does, however, reach an average of over 98%

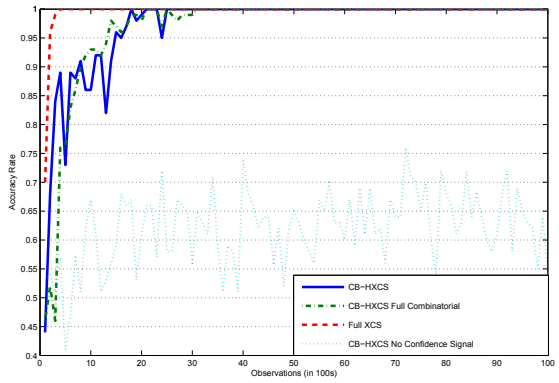

Figure 5: Multiplexer-6: CB-HXCS compared to Full XCS

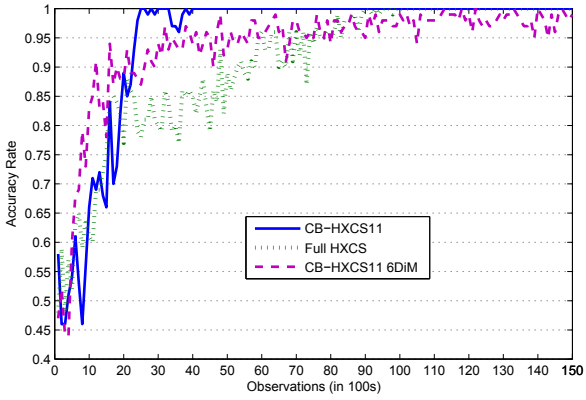

Figure 6: Multiplexer-11: CB-HXCS compared to Full XCS

accuracy. In addition, it reaches 90% in only 2,500 observations, compared to only 80% at that step for the full XCS (due to the greater complexity of this model, the exploit rate parameter was decreased to ($\text{rate}_t = .9997 * rate_{t-1}$)).

5. CONCLUSIONS AND FUTURE WORK

The application of the CB-HXCS to the multiplexer data generated the following conclusions:

1. Voting margins can be predictive of accuracy rates for the XCS over different partitions of the problem space
2. It is possible for the CB-HXCS method to solve classification problems to a similar degree of accuracy as an XCS with full access to the environmental data
3. In certain situations, the CB-HXCS uses data more efficiently, solving classification problems with far less data

Voting patterns of homogeneous learning classifiers can provide information about the ability of the classifier to accurately classify a given state. This reflexive capability has also been shown to improve agent communication by enabling the XCS agent to signal to the network when it believes that it is able to make sound classifications. The hierarchical network provides a method to reassemble the results of the sub problems in order to solve the global problem. The improved agent signaling allows for efficient communication with in the network.

Both of the multiplexer problems were solved by the CB-HXCS. Not surprisingly, at the larger task, the CB-HXCS outperformed the full XCS method, if performance is measured in the quantity of data consumed. As the relative size of the original environment increases with respect to the partitions, it is expected that the CB-HXCS will improve its relative performance.

There are drawbacks of the CB-HXCS method:

1. It requires that the problem can be separated into smaller subproblems. For problems with rules that require the use of all of the dimensions to classify, the CB-HXCS is likely not to be appropriate.
2. Another practical problem is that while it is possible to solve the global problem by adding the results from smaller subspaces, the total number of agents needed balloons as the original dimension increases. If only the maximal rule length is assumed and there is no other information to partition the data, then the number of base level agents required grows to n!/(r!*(n-r)!), where r is the maximal rule length and n is the number of original dimensions. In situations were data acquisition is relatively expensive it may be appropriate to solve the problem using the full combinatorial approach. A possible way around the full combinatorial approach is to trade off a 'small' increase in partition size so that fewer agents are required to span all possible rules of a given length. Our early results are promising but still need further investigation. However, even the 8-4-2-1 network for the multiplexer-11 problem required the use of 3,015 (15*201) individual XCS processes.
3. The interpretation of the learned rules is not as clear as in a standard XCS. Not only are the rules based on the network hierarchy, but there are many classifiers per agent (one set per micro agent).

Future work will focus on larger and more diverse problem domains, the interplay between the various XCS parameters, and the possible application of the to multi-class problems.

6. REFERENCES

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[2] M. V. Butz. XCSJava 1.0: An Implementation of the XCS classifier system in Java. Technical Report 2000027, Illinois Genetic Algorithms Laboratory, 2000.

[3] H. H. Dam, H. A. Abbass, and C. Lokan. Dxcs: an xcs system for distributed data mining. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1883–1890, New York, NY, USA, 2005. ACM Press.

[4] T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[5] S. W. Wilson. Classifier Systems and the Animat Problem. *Machine Learning*, 2:199–228, 1987.

[6] S. W. Wilson. Classifier Systems Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[7] S. W. Wilson and D. E. Goldberg. A Critical Review of Classifier Systems. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 244–255. Morgan Kauffmann, 1989.

[8] D. H. Wolpert. Stacked generalization. Technical Report LA-UR-90-3460, Los Alamos, NM, 1990.