XCS and the Monk's Problems

Shaun Saxon
The Database Group,
Colston Centre, Colston Avenue
Bristol, UK.

Email : Shaun.Saxon@uwe.ac.uk
Phone : (++44) 117 965 3607

Alwyn Barry
Faculty of Computer Studies and Mathematics,
University of the West of England,
Coldharbour Lane, Bristol, BS16 1QY, UK

Email: Alwyn.Barry@uwe.ac.uk
Phone: (++44) 117 965 6261 ext. 3777

Abstract

It has been known for some time that Learning Classifier Systems (Holland, 1986) have potential for application as Data Mining tools. Parodi and Bonelli (1990) applied the Boole LCS (Wilson, 1985) to a Lymphography data set and reported 82% classification rates. More recent work, such as GA-Miner (Flockhart, 1995) has sought to extend the application of LCS to larger commercial data sets, introducing more complex attribute encoding techniques, static niching, and hybrid genetic operators in order to address the problems presented by large search spaces. Despite these results, the traditional LCS formulation has shown itself to be unreliable in the formation of accurate optimal generalisations, which are vital for the reduction of results to a human readable form. XCS (Wilson, 1995, 1998) has been shown to be capable of generating a complete and optimally accurate mapping of a test environment (Kovacs, 1996) and therefore presents a new opportunity for the application of Learning Classifier Systems to Data Mining. As part of a continuing research effort this paper presents some first results in the application of XCS to a Data Mining task. It demonstrates that XCS is able to produce a classification performance and rule set which exceeds the performance of most current Machine Learning techniques when applied to the Monk's problems (Thrun, 1991).

1 INTRODUCTION

With the increase in the computerization of business, and in particular retail business, large amounts of business data is gathered and stored. Until recently the benefit of attempting to extract useful information from this data only marginally outweighed the cost of maintaining this great a volume of data because the then current data analysis techniques were ill-suited to coping with large and noisy data and were not able to extract information of sufficient quality and quantity. With the reduction in the cost of storing data, there is now pressure to effectively utilise the data for the benefit of the data producers. For example, improvements in the performance of data analysis techniques would be beneficial for a large retail chain with computerized stock control and sales systems by enabling it to analyze its large databases to predict customer behavior. Furthermore, the recent popularity of loyalty cards generates customer specific transaction data, which increases the potential for forming more accurate models of customer behavior.

Current techniques for discovering patterns in data, termed 'Data Mining', have various limitations when considered against criteria for an effective data mining method. These criteria include the ability to cope with large and noisy data, being able to produce simple, meaningful and useful results that can be understood by a human user, and the use of a practical amount of computational resources. Evolutionary computation techniques, including those based on genetic algorithms, are able to explore a large search space to find an optimal solution effectively due to their implicitly parallel nature. They are also [largely] undirected and therefore require minimum input of domain knowledge. These robust techniques have therefore been considered in order to counter the limitations of current data mining techniques and have already been successfully applied in a commercial environment (Flockhart, 1995).

Whilst holding much promise, there are also known difficulties with the application of GA based techniques to the task of Data Mining. Traditional Learning Classifier Systems (Holland, 1986) which can be used for this purpose have a limited ability to generalize rules. A traditional LCS provides generalization pressure by the use of specificity measures that are factored into the action selection process (Goldberg, 1989; Riolo, 1987). This method does not produce optimal generalization, tending to over generalize in the early stages of learning, only to over specialize as the GA continues to operate (Riolo, 1988; Wilson, 1988). Wilson's XCS (1995, 1998) introduced niching techniques and an accuracy-based fitness which enables XCS to produce accurate optimally general co-operative rule sets. This ability to generalize lays the foundations of a concept builder of the kind required for data mining.

In investigating the performance of a Data Mining algorithm a comparison with the performance of existing techniques is useful. The Monk's Problem is one of only a few classification problems that has been used to generate a substantial comparative study (Thrun, 1991). As a first investigation of the abilities of XCS as a Data Mining tool the use of the Monk's Problem is appropriate, providing strong evidence of the qualities of XCS classification in comparison to other machine learning techniques. This paper details these investigations and seeks to identify areas of strength and weakness in XCS for further ongoing research into the application of XCS to the Data Mining of large commercial data sets.

2 BACKGROUND

This potential for extraction of useful trends and indicators from consumer data can be realized providing that the problems of increased database size can be overcome by improved data analysis and reduction techniques which are constituents of the larger process commonly known as Knowledge Discovery in Databases (KDD) (Fayyad, 1996). This term is used to describe all of the tasks required to extract useful information from raw databases and to distinguish the task of automatically discovering patterns in pre-processed data, Data Mining, from data extraction, data cleansing, data reduction, and interpretation and application of results from data mining. Within the context of Data Mining, classification is the process of using a set of pre-classified instances to assign a new instance to one of *n* classes. A finite vector of attribute values defines an instance. The training set is a set of instances that have been assigned to one of the *n* classes.

Current data mining techniques (Saxon, 1998) have been developed from methods originating in the fields of statistical modeling, neural networks and machine learning. Statistical techniques (Mitchell, 1994; Molina, 1994) exist for classification (e.g. CART (Breiman, 1984) and CHAID (Kass, 1980)), regression (linear, quadratic and logistic) and clustering tasks but are based on assumptions about the probability distributions of data, which limits their effectiveness on real data, and can only be applied with the intervention of a statistician. Advanced machine learning techniques have been applied to data mining tasks with varying degrees of success. Decision tree (ID3, C4.5) and production rule (AQ15, CN2) learners[1] produce fairly accurate but complex representations of patterns in the data although these representations may be simplified by using heuristic search. Generalization occurs when grouping attribute values and when decision trees are pruned. These techniques often require considerable computation and can be applied to limited data types therefore much effort needs to be put into data reduction prior to application. Artificial neural networks have been used widely with considerable success for Data Mining tasks (Lu, 1995; Rowher, 1991) and are implemented in many commercial data mining software packages. They are able to learn complex models of data and provide good generalization but the interpretation and application of the neural network models are difficult tasks for the human user.

In order to counter the problems with the current data mining techniques there have been projects which sought to apply evolutionary computing techniques to various areas within KDD. Evolutionary techniques have primarily concentrated upon the KDD task of Feature Selection (e.g. Punch, 1993), and advances have been made in the modification of GA techniques to develop and maintain both single (GABL, GABIL, (Spears, 1992)) and multiple (COGIN, (Greene 1993, 1994) target concepts. Recent work within Feature Selection has moved onto the application of Genetic Programming with pruning techniques introduced to produce human readable results (eg. Raymer, 1997). Applying evolutionary computing to Data Mining is a more complex task, directly tackling the learning problem rather than metalearning. Early work by Parido and Bonelli (1990) with the Animat LCS (Wilson, 1983) illustrated the potential of LCS for Data Mining within a simple data mining problem, and two significant research projects have sought to extend these results to commercial data sets (see REGAL (Giordana, 1994) and GA-Miner (Flockhart, 1995)). Significantly, both of these projects introduce high level attribute and attribute vector encoding with directed mutation and crossover operators in order to cope with the large search space presented by commercial data sets. Problems with these approaches remain, however. In particular, with GA-Miner for example, the establishment and maintenance of multiple co-operative rules in the presence of the GA has been tackled using static techniques that can require domain knowledge, and the methods applied to obtain generalization are primitive. These problems are common to all traditional Michigan style LCS implementations.

The XCS Classifier System (Wilson, 1995, 1998) represents a major step forward for Michigan style LCS. Deriving from Wilson's Animat (Wilson, 1985) and ZCS (Wilson, 1994), the XCS simplifies the Holland style classifier system in an attempt to produce a LCS whose operation and dynamics can be readily understood and predicted. It borrows strength update mechanisms from Reinforcement Learning (Watkins, 1991) using a modified Widrow-Hoff update mechanism to provide a multi-parameter strength regime that more accurately reflects the different roles of strength within action selection and the GA. XCS also re-introduces GA mechanisms recommended by Booker (1989) which provide a niching facility to allow co-operative sets of rules to co-exist within a population whilst encouraging competing rules to converge on optimum rule attributes within a niche. By using accuracy as the GA selection criteria, XCS is the first LCS to be able to claim to 'reliably generate the most general accurate classifiers' - the so-called 'Generalization Hypothesis' (Wilson, 1998). Further work by Kovacs (Kovacs, 1997) has demonstrated that the provision of further operators can sufficiently focus the population once exploration is complete to reliably produce a minimum rule set consisting of the most general accurate classifiers, and has led to the 'Optimality Hypothesis' which suggests that

[1] For further information on these and similar techniques Mitchell (1997) provides an accessible though detailed review chapter.

using these operators the rule set generated will be the optimal rule set for a given static problem.

It would appear, therefore, that the application of [a possibly modified form of] XCS to Data Mining has the potential to address the significant problems which remain unsolved within previous Michigan style LCS based Data Miners. As part of a continuing project to develop an effective Data Mining tool using Evolutionary Computing techniques, an investigation into the potential of XCS is underway. Learning from previous efforts, it is expected that XCS will need to be modified to accommodate more complex attribute and attribute vector encoding and that hybrid genetic operators will be applied. Before venturing further in these investigations, however, it is important to identify a baseline performance of XCS in order to ascertain its appropriateness for the task of Data Mining. It was decided that XCS should be applied to a small data set in order to demonstrate the effectiveness of its generalization and niching mechanisms within the domain of Data Mining.

3 EXPERIMENTAL INVESTIGATION

3.1 TEST ENVIRONMENT

For an initial investigation of the suitability of XCS to the task of Data Mining the use of a 'standard' test data set was considered important. Unfortunately within the field of Data Mining no such test sets exist, with the best attempt at a standard test set being the data sets used with the STATLOG project (Michie, 1994) which due to their complexity and quantity are inappropriate for use at this stage. Within the Machine Learning community a number of data sets have a sufficient usage to make them a 'de facto' standard. The Monk's test set, held on the UCI Repository {REF}was selected as an appropriate test set because of its small size with high internal complexity, and the extensive set of published comparative results available with the test data.

The Monk's problems are binary classification problems of differing complexity based on a pre-classified artificial robot data set created to empirically compare learning classifiers (Thrun, 1991). There are three problems each of which involves learning target concepts described by conjunctions of logical relations on attributes (see Thrun). Training and test data sets are created by partitioning the data set containing all possible instances and pre-classifying each instance according to the problem. 5% noise is added to the training data for the third Monk's problem. Objective assessment of the performance of XCS is feasible because the data has been pre-classified according to explicit and well-defined target concepts which enables a direct comparison to be made with the performance of other Data Mining techniques on these problems.

The three test sets provided with the Monk's problem are individually significant to the evaluation of XCS for Data Mining. The first test set is solved using a simple boolean expression which, whilst not directly representable in a single XCS classifier condition, can be optimally represented (including the corresponding negative cases) using the standard XCS ternary conditions by small set of 12 cooperating classifiers. Four of the six attributes must be removed from consideration using generalization, thereby providing a test of the generalization capabilities of XCS. The second experiment requires a complex set of disjunctions of conjunctions to be fully represented, and presents a considerable challenge to most Machine Learning techniques. This experiment will test the ability of XCS to maintain a population of many cooperative classifiers in order to represent the classification accurately. In real data it is difficult to clean the data completely, and the use of accuracy as the basis of XCS learning could be compromised by noisy data. Thus, the Monk's 3 data set will illustrate the ability of XCS to cope with typical amounts of noise found in data after data cleansing without hindering the learning operation.

3.2 THE LEARNING SYSTEM

A XCS implementation conforming to the XCS used within (Wilson, 1998; Kovacs, 1996) was obtained (Barry, 1998). In all experiments the parameterization of the XCS was set as follows: p_1 =10.0, ε_1=0.01, F_1=0.01, β=0.2, γ=0.71, θ=25, ε_0=0.01, α=0.01, χ=0.8, μ=0.04, ϕ=0.5, $P_\#$=0.33 (see Kovacs (1996) Appendix B for a parameter glossary). These parameter settings were taken from the settings used by Wilson (1995), and were considered appropriate due to the similarity of the Monk's search problem to the Boolean Multiplexor problems investigated by Wilson. During our investigations alternative parameter settings were investigated and not adopted. The message length, condition length, and maximum population size were set according to the needs of each experiment.

The Monk's problems are derived from a data set of 432 instances, each consisting of six categorical integer valued attributes (see Thrun). Initially the input encoding was devised as a binary string with each attribute represented by a minimal corresponding binary form concatenated to produce the input bit vector. For example, attribute 1 is a three valued attributed (values 1, 2, and 3), represented in this encoding by two bits with value 1 coded as 00, value 2 coded as 01, and value 3 coded as 10. This gives a string length of 10 to represent all six attributes. The minimum string length required to represent the entire attribute space of 432 is 9 (2^9 = 512). Thus, this representation produces a small degree of redundancy. The encoding assumes that the attribute values are ordered in some way, even arbitrarily. This assumption is satisfied in the Monk's data because the categorical attribute values are represented as successive integers starting at one. The output message is a single binary bit representing the predicted class for instances matching the classifier condition.

A training and test environment was constructed to present the data instances sampled at random from the training set on each iteration of the XCS. Upon receiving the chosen output message from the XCS, the environment returns a maximum reward if the output message is the same as the class of the instance used to provide the input message and a minimum reward is returned otherwise. To test the learnt classifier population, the test set is presented to the XCS after a predetermined number of exploration (learning) trials, set as appropriate for each experiment, and the performance of XCS captured during this phase with classifier induction and strength modification disabled. The results presented are an average of 10 runs of XCS on the training set and with constant parameterization. The results of this test phase are presented only if the performance of the XCS on the test data set was different from that achieved within the training set at the end of the learning phase. Coverage Tables, used to compare the completeness of concept learning within Thrun (1991), were also captured at the end of the learning phase.

The graphs show three measures of XCS performance: system prediction, system error and macroclassifier population size. System prediction indicates the proportion of exploit iterations, out of the previous 50 exploitation iterations, in which XCS correctly classifies a data instance. The system error is a moving average of the error in the prediction of the chosen Action Set [A] (the set of classifiers chosen to perform the action in an iteration), again calculated over the previous 50 exploitation iterations. The population size is the proportion of the number of macroclassifiers in the population to the total number of microclassifiers that the population may hold. For an XCS that successfully learns an optimal set of classifiers the prediction rises steeply to 1.0 and stays at that level. Occasional fluctuations just below this maximum may occur when the GA deletes a classifier during search for the optimal population (termed [O]; (Kovacs, 1996)). Macroclassifier population size will typically increase to a peak at around three-quarters of the total population size as the XCS explores as many classifiers as possible. The pressure to obtain optimally general classifiers drives the population down as the “good” classifiers increase in numerosity and the “poor” ones are deleted. Eventually the population will stabilize when the XCS has found the optimal set. A sign of a weak performance within XCS (when it doesn't explore a high enough proportion of the population and hence is unable to converge on the optimal solution) is when the population size oscillates around a relatively high proportion (> 0.3) without further reduction. Note that the proportion of macroclassifiers in the stable population depends on the population limit set as a parameter and that this limit is set using knowledge of the domain.

3.3 THE MONK'S 1 PROBLEM TEST

The Monk's 1 test is a relatively simple test designed to demonstrate that a new classification algorithm is capable of simple concept learning. As such, the application of this test set to XCS should confirm or deny the basic hypothesis that XCS is an appropriate basis for the development of a Data Mining tool. From results reported for the testing of other Machine Learning algorithms on this test set (Thrun, 1991) it would be expected that XCS should be able to demonstrate accurate classification (demonstrated by accurate prediction) within a relatively short learning period. In addition, it is desirable that XCS be able to identify the optimal classifier set for this problem. The optimal set of twelve classifiers for the Monk’s 1 problem using binary encoding is: {#00#######→1, ######0000→1, #######1#1→1, ######1#1#→1, #1####0#1#→0, ##1###0#1#→0, #1####1#0#→0, ##1###1#0#→0, #1#####0#1→0, ##1####0#1→0, #1#####1#0→0, ##1####1#0→0}. Note that XCS retains accurately wrong classifiers therefore each of these classifiers will have an associated classifier with the same condition, the opposite action, and a prediction equal to the minimum reward value.

The condition size and message size for the binary encoding used with this test were set at 10. The population size limit was chosen from a consideration of the predicted size of [O]. If we want to find all the optimal classifiers each with an average numerosity of fifteen then the minimum population size should be 24 . 15 = 360 micro classifiers. Thus, for this experiment a population size of 400 was deemed adequate.

Results

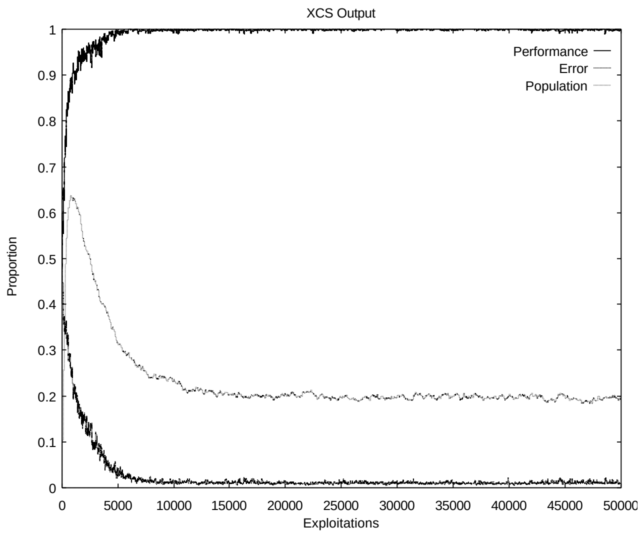

Figure 1 : XCS performance during training using the Monk’s 1 Test data set, averaged over 10 runs with parameterization as described in Section 3.2. The performance during the test set remained at the same level as the final 5,000 iterations.

The performance measures averaged over 10 runs of XCS on the first Monk's training set indicate that XCS had learned the target concepts after only 10,000 exploitations. The performance curve in Figure 1 indicates that XCS has successfully established a population that correctly classifies the data set. An analysis of the populations resulting from the individual runs revealed that all of the populations had discovered [O] with the members of [O] obtaining high numerosity, accurate prediction and fitness. In three of the final populations one of the optimally general classifiers did not have a relatively high numerosity or experience but was still perfectly accurate and had a high fitness, demonstrating that this final member of [O] had been established relatively late within the run in these cases. On average each final population held 80 macroclassifiers, 24 of which made up [O]. All of the remaining classifiers had low numerosity (less than 5) and small experience, indicating that they were the product of continued but unnecessary GA search.

These results confirm the hypothesis that XCS is able to induce and establish the accurate and optimally general rule set from a simple data set of pre-classified data. The performance of XCS was comparable with the performance of neural networks and production rule classifiers (CN2 and those based on AQ-15), and exceeded that of all the ID3 based decision tree classifiers reported in Thrun (1991).

3.4 MONK'S 2 PROBLEM TEST

The Monk's 2 data set represents a much more complex relationship than that seen in Monk's 1. In the comparison of performance (Thrun, 1991) only the neural network based classifiers and AQ17DCI obtained 100% classification performance, with most decision tree classifiers obtaining 65-70% correct classification. The complexity of the task is indicated by the fact that in order to represent the concepts optimally within the ternary condition encoding of XCS a set of 104 binary-encoded classifiers is required for positive classification and 100 classifiers for negative classification. It must be noted, however, that the target concept in Monk's 2 Problem cannot be simply described when using disjunctions of conjunctions of attribute-value pairs and therefore this problem is artificially hard for XCS using the standard condition representation.

The XCS message encoding used within this test was initially kept the same as that used for the Monk's 1 test. Knowing the size of the solution space, however, the population size was increased to 1600 classifiers. This is potentially small for the predicted number of target classifiers but investigations into larger population size did not show improvements sufficient to justify the extra computational resources required.

Results

It can be seen from the XCS performance in Figure 2 that it struggles to find the optimal set of classifiers. The system prediction converges slowly towards 1.0 but still averages only about 0.98 at 50,000 exploitations. The proportion of macroclassifiers in the population peaks at only 0.6 and then falls to ~0.35 over the whole 50,000 exploitations, indicating that the XCS has not explored enough of the solution space to find all the optimally general classifiers. System error stabilizes at about 0.05 which is significantly larger than the accuracy cutoff of 0.01 supporting the suggestion that insufficient search has been performed. Inspection of the final populations and the corresponding coverage tables revealed that XCS had only learnt a small proportion of the optimal rule set in each of the runs if only the high numerosity classifiers are considered. A larger proportion of the optimal rule set was present if ‘weak’ classifiers were considered.

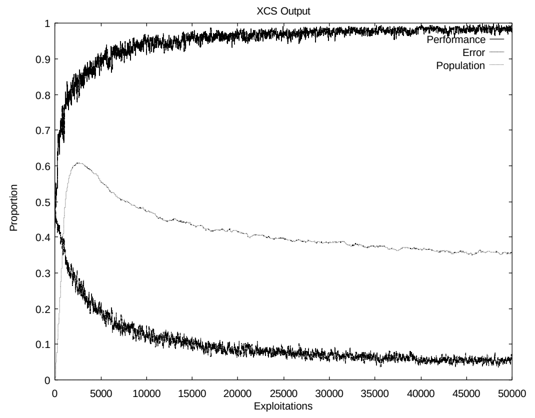

Figure 2 : XCS performance in the Monk’s 2 Test data set using a Binary Encoding, averaged over 10 runs, population size 1600.

From these results it was hypothesized that XCS was unable to find suitable classifiers due to the deceptive nature of the encoding used. To test this hypothesis a second encoding type was introduced - an enumeration encoding. The enumeration encoding maps an attribute with *n* possible values to a binary string of length *n*. The attribute value is mapped to an integer ranking, *i*, by setting the i^{th} lowest bit to 1 and all the others to 0. This gives a string length of 17 for the Monk's Problems and a condition search space of 3^{17}. In fact, using this encoding in XCS allows the same attribute vector to be encoded in more than one way. For example, the attribute-value pair (A1, 3) can be represented using the ternary strings, '100', '1#0', '10#', '1##', and '#00' because (A1, 3) can only ever be encoded in a binary string as '100' in an input message. Also, it is more convenient to express inequalities such as A1≠1, which in an enumeration encoding has a single representation '##0' whereas in a binary-valued encoding two strings are required: '1#' and '01'. This reduces the number of classifiers required from 204 for the binary encoding to 42 - 15 and 27 classifiers for positive and negative classification respectively. It was expected that the greater flexibility of the enumeration encoding in representing attribute-value relations will enable XCS to optimally classify for the Monk’s 2 Problem and to find the optimal rule set.

The encoding was adjusted within the test and trial environment and the experiment was repeated with the new encoding. The population size was maintained at 1600 microclassifiers in order to allow a direct comparison with the results from the binary encoding.

The averaged results of the 10 runs are depicted in Figure 3. With the enumeration encoding the system prediction reaches its maximum of 1.0 at around 25,000 exploitations with the system error correspondingly dropping to around 0.0, as expected. This demonstrates that XCS was able to identify a correct classification for the test data set. An examination of the final populations revealed that [O] was not completely formed, with approximately 30 of the 42 members of [O] present and the remainder covered by two or more classifiers with conditions that remained too specific.

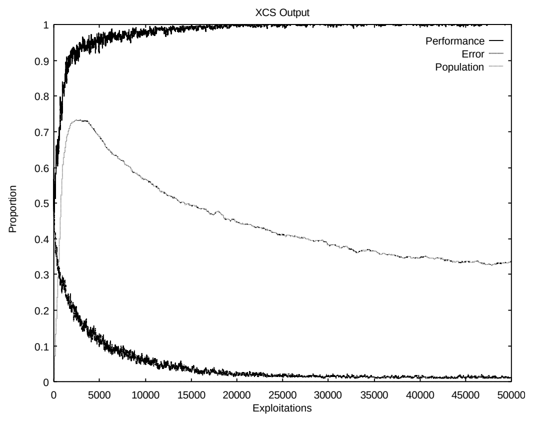

Figure 3 : XCS performance in the Monk's 2 Test data set using the Enumeration Encoding (averaged over 10 runs, population size = 1600), demonstrating the ability of XCS to perform optimally given a suitable classifier encoding.

These results confirmed the hypothesis that XCS was able to learn the correct classification of a data set representing a complex concept. Clearly this performance is an improvement over most Machine Learning algorithms reported in Thrun (1991). The results reported for the application of Back-Prop neural networks could not generate generalized concepts that were open to examination, which is a key requirement of an effective Data Mining technique.

3.5 MONK'S 3 PROBLEM TEST

The target concept of the Monk's 3 Problem is of similar complexity to Monk's 1 but it incorporates noise by including 6 misclassified examples (~5%) in the training set. This test data set could pose significant problems for XCS, with the ability to maintain 'strong' classifiers dependent upon the value of the parameters of the accuracy function. Data that contains too much noise could be preventing XCS from forming accurate generalizations and thereby hinder the search process of the GA within XCS. Alternatively, where a lower level of noise is present XCS may over generalize ('overfit' to the training data), hindering performance on the test set. It is hypothesized that the limited extent of noise added to the Monk's 3 data set will not hinder the ability of XCS to establish the best [O] available from the training set, although XCS may overfit to the training set.

This experiment was run using 10 runs in each case of both the Binary encoding and the Enumeration encoding, with the condition and message size set appropriately for each encoding. The population size was set to 400 because of the lower level of complexity of this problem, requiring 22 classifiers to map the [noiseless] problem completely with optimal generality when using the Binary encoding.

Results

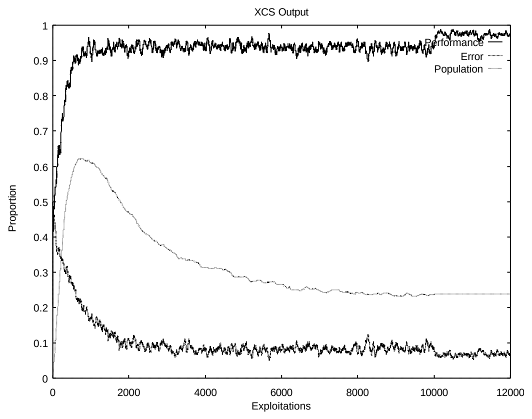

Figure 4 : XCS performance in the Monk's 3 Test data set (averaged over 10 runs, population size = 400). The last 2000 exploration iterations were conducted without the induction algorithms using the Test data set. The optimal expected performance was 95% due to the noisy training set.

No difference in performance was found as a result of differences in the encoding form used. Figure 4 shows the results of running XCS using the Binary encoding on the Monk's 3 training and test data sets. Training XCS gives a classification performance of ~0.95 and a relatively large system error. This is a consequence of the noise in the training set, with a maximum achievable performance of 0.95 on average. When XCS is switched to run on the test data set the classification performance improves and the system error drops. An examination of the populations at the end of each run revealed that XCS had successfully established a set of high numerosity optimally general classifiers close to [O] (which will never be achieved because of the presence of noise). We hypothesize that the improvement in performance during the test phase is due to the presence of other classifiers in the population as a result of continued GA exploration which cover the over generality due to noise of the high numerosity classifiers.

The results achieved by XCS are comparable with those obtained by Neural Network classifiers in Thrun (1991), and marginally better than the Decision Tree classifiers. They were not as good as the AQ15 based Production Rule classifiers which typically achieved 100% performance in the test set due to the larger (normally redundant) coverage of the classification space which they maintain. These results are encouraging, although staged experiments introducing graduated amounts of noise are required to fully evaluate the ability of XCS to cope with noisy data.

4 DISCUSSION

The performance of XCS within the Monk's problems is very encouraging. Comparisons with existing Machine Learning techniques have already been made in Section 3, which demonstrate that the performance of XCS is at least as good as, and in many cases better than classifiers based on other Machine Learning techniques. Furthermore, the ability of XCS to determine the compact rule representation provided by the optimally general classifier set in these tests, and potentially present these classifiers in a human readable form, is a clear advantage.

In the Monk's 2 Problem the XCS did not form a completely generalized mapping of the target concepts. In these populations a number of classifiers existed which were marginally over specific, indicating that more GA recombination was required in order to obtain full generalization. Kovacs (1997) has introduced additional deletion operators in a mechanism termed "condensation", triggered to reduce a population to the optimal set. It is likely that a mechanism of this type could be employed to enable XCS to reliably produce a minimal set of optimal rules even when learning is halted prior to establishing [O].

In data mining applications the target concepts are rarely known, although sometimes estimated by a domain expert, so we have resisted the temptation to introduce any encoding that is tailored to the problem. At least one of the production rule algorithms compared in the Thrun paper (Bala, 1991) produces 100% accuracy on both the Monk's 2 test sets. This is wholly due to the fact that the algorithm had been set to learn concepts of based on the number of first-valued attributes, which enables the target concepts to be represented by only two production rules:

```
if  (equals (number of attributes that are first valued, 2)) then 1
if  (notEquals (number of attributes that are first valued, 2)) then 0
```

Without specific knowledge of the target concepts the classifier system cannot be setup to use rules such as these so more general representations are needed. For these artificial problems the target concepts are known and therefore an encoding that is concept-specific would theoretically perform better on this problem than the more general ternary encoding used.

Theoretically, the enumeration encoding should be used for all of the Monk’s problems because all of the attributes are categorically valued and the binary encoding becomes unwieldy when relations between attributes and logical negations need to be represented. Practically, because the enumeration encoding is longer than the binary for the Monk’s problem and there are many more redundant classifiers than when learning relatively simple target concepts such as in the Monk’s 1 problem, the compactness of the binary encoding makes it just as, or more, effective than the enumeration encoding. These encodings are suitable only for discretely-valued attributes so future work will include investigation into encodings for continuously valued attributes and alternatives encodings for discrete attributes. It is recognized that GA-Miner and other LCS based Data Mining tools which have been developed have employed high level attribute representations and more complex classifier condition formulations which are advantageous in many circumstances. Future work will address these issues.

The difficulty in learning the concepts for the Monk's 2 training set using the Binary encoding require further investigation. Although intuitively XCS would be expected to increase its level of exploration when correctly classifying rules cannot be found, the degree of exploration used by XCS was actually reduced. This would suggest that the GA in XCS was unable to recombine classifiers in a way that produced significant improvements in their accuracy, so preventing a clear search direction emerging which would be exploited by the GA. This may indicate a potential problem within XCS, or it may be the case that the binary encoding simply presented a deceptive landscape. Further investigation of this phenomenon is required.

An important criterion for an effective data mining technique is the ability to cope with noisy data. The large number of samples taken from only 124 training examples presented the danger of over-fitting to the training data and succumbing to bias within the training data. The tension between obtaining accurate classification and achieving maximally general classifiers in the operation of XCS appeared to prevent this. More investigation is required in order to ascertain the proportion of noise (and/or bias) in the data which XCS is able to absorb before performance degrades significantly.

5 FUTURE WORK

The Monk's problems are based on a data set with six attributes each having no more than four values making a total of 432 possible attribute vectors. A typical commercial data set that has been prepared for analysis will have up to 100 ordinal and nominal, continuous and discrete attributes and has a dimensionality several orders higher than the Monk's data sets. The results presented in this paper have demonstrated that XCS classifies the Monk's data sets more accurately than many current classifier systems, and at least as well as those systems based on artificial neural networks,. In order to establish the effectiveness of XCS and GA-based classifier systems as Data Mining techniques, then the performance of XCS on the large and noisy commercial data must be investigated.

XCS employs the ternary alphabet which is used in the canonical LCS and which grew out of the binary encodings used in GAs. The results in this paper demonstrated that even when restricted to this alphabet, changing the encoding causes an observable change in the performance of XCS and is itself affected by the nature of the concepts to be learnt. An effective encoding must be able to compactly and accurately represent patterns in data, of which there is typically little prior information, whilst keeping disruptions of the generalisation mechanisms of XCS to the minimum. There has been much work (Goldberg,) on representing discrete and continuous values in a GA but these representations are not always adequate for a classifier. For example, standard binary-based representations for many-valued discrete attributes will produce prohibitively long and inefficient bit strings if there are several of these attributes that are sparsely-valued. Coping with the uncertainty of noisy data will require fuzzy or windowed representations (Booker,). Mixing representations in a single classifier has implications for the crossover operators used by XCS and thus the correct operation of XCS niching and generalisation. Thus, any new or modified representations will necessitate creating original operators and investigating their effect on XCS performance.

Establishing a set of effective encodings will enable XCS to be tested over a wider range of test data sets and allow a more detailed comparison of its performance against other classifier systems.

The performance of XCS on the data sets used to compare Machine Learning, statistical and Artificial Neural Network techniques in the Statlog project (Michie et al) would provide a good measure of XCS's relative effectiveness as a classifying system. Several of the data sets used within the Statlog project were taken from real commercial sources and are characteristically large and noisy and would therefore be appropriate performance reference points. For effective comparison between Data Mining investigations within the LCS community it is vital that these common data sets are adopted.

XCS has the ability to identify the attributes that have a significant effect on the classification by completely generalising the contribution of the less important attributes. There will be a practical limit on the number of attributes that XCS can handle which will be exceeded by many commercial data sets so it is important to apply suitable feature selection techniques. One such technique, taken from statistical modelling, is to increment or decrement the number of attributes depending on the classification performance of the preceeding model generated by the Data Miner and the relative value of each particular attribute in this model. Statistics on the distribution of attribute values for each class are used to measure of the importance of an attribute. XCS provides an excellent platform for integrating these mechanisms dynamically so that attributes can be added and subtracted in a single run of XCS.

6 CONCLUSIONS

In the application of the Monk's problems to XCS is has been shown that XCS was able to perform at least as well as traditional Machine Learning techniques in both simple and complex classification tasks, and in the presence or absence of small proportions of noise within the data. In some cases XCS was seen to perform better than Machine Learning techniques which are commonly found in commercial data mining applications. XCS was also able to produce and maintain an easily identifiable accurately general set of classifiers representing the concepts within the data sets. If combined with software to convert the ternary classifier representation into a standard production rule format, XCS will generate a model that can be readily understood and reapplied by potential clients of a data mining application.

Whilst encouraging, the Monk's problems are limited in their correlation to data sets used within a commercial Data Mining environment. Work is underway on expanding the application of XCS to graded real world data sets that will allow the XCS mechanism to be scaled to cope with large or sparse data sets. In particular, the encoding problems seen within the Monk's 2 problem indicate a need to find appropriate attribute representation, and appropriate condition representations. Large data will also require more input from hybrid genetic and non-genetic operators that will take into account knowledge about the distribution of attributes within the data set. It is also possible that methods akin to those used to establish linkage within genotypes may bear fruit, and this avenue will be investigated.

If GA-based Data Mining is to compete with the statistical modelling, artificial neural networks and decision-tree learners that are well-established Data Mining techniques then it must at least be able to offer the ability for supervised classification of large and noisy data over many different domains. Of the previous work in GA-based techniques for Data Mining outlined earlier only REGAL and GA-Miner have been used successfully to mine commercial data by using hybrid techniques to solve some of the practical problems of mining commercial data but these use domain knowledge and primitive generalisation operators. XCS has the potential to extend the capabilities of GA-based data mining systems over and above other Data Mining techniques because it has the ability to produce the optimally general descriptions of patterns in any data. This, together with the robust nature of GAs, will mean that a GA-based data mining system with XCS capabilities will have an accuracy comparable with that of Neural Networks, and will provide an explicit human-readable description of the patterns in the data equivalent to those produced by decision-tree learners.

Acknowledgements

Project funded by UK Department of Trade and Industry under the auspices of the Teaching Company Directorate.The authors wish to acknowledge the advice and discussions with Stewart Wilson and Tim Kovacs in the development and testing of the XCS implementation. They also acknowledge the help, support, and resources given by The Database Group and Dr Andrew Greenyer, in the conduct of this project.

Bibliography

Bala, J., Bloedorn, E., Kaufman, K., Michalski, R.S., Pachowicz, P., Vafaie, H., Wnek, J., Zhang, J. (1991), Applying Various AQ Programs to the MONK’s Problems: Results and Brief Description of the Methods, *in* Thrun (1991), 7-22.

Barry (1998), The XCS Classifier System, *Tech. Rep.*, Faculty of Computer Studies and Maths, UWE, UK.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984), *Classification and Regression Trees*, Wadsworth

Booker, L.B. (1989), Triggered Rule Discovery in Classifier Systems, *in* Schaffer, J.D. (ed), *Proc. Third Intl. Conf. On Genetic Algorithms*, 265-274, Morgan-Kaufmann.

Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth, P. (1996), From Data Mining to Knowledge Discovery: An

Overview, *in* Fayyad, U.M., Piatetetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (eds), *Advances in Knowledge Discovery and Data Mining*, AAAI Press.

Flockhart, I. (1995), GA-Miner: Parallel Data Mining and Hierarchical Genetic Algorithms, *Tech. Rep.: EPCC-AIKMS-GA-MINER-REPORT 1.0*, Univ. of Edinburgh.

Giordana, A., Neri, F., Saitta, L. (1994), Search-Intensive Concept Induction, *Tech. Rep.*, Universita di Torino, Dipaertimento di Informatica, Torino, Italy.

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

Greene, D.P., Smith, S.F. (1993), Competition-based Induction of Decision Models from Examples, *Machine Learning*, 13, 229-257.

Greene, D.P., Smith, S.F. (1994), Using Coverage as a Model-Building Constraint in Learning Classifier Systems, *Evolutionary Computation*, 2(1).

Holland, J.H. (1986), Escaping Brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems, in M. Michalski, M. Carbonell (Eds), *Machine Learning, an Artificial Intelligence approach*, *Vol. 2*, pp. 593-623, Morgan Kauffman.

Kass, G.V. (1980), An exploratory technique for investigating large quantities of categorical data, *Applied Statistics*, 119-127.

Kovacs, T. (1996), Evolving optimal populations with XCS classifier systems, *Tech. Rep.: CSR-96-17*, School of Computer Science, University of Birmingham, UK.

Kovacs, T. (1997), XCS Classifier System Reliably Evolves Accurate, Complete and Minimal Representations for Boolean Functions, *WSC2: 2nd On-line World Conf. on Soft Computing in Engineering Design*.

Lu, H., Setiono, R., Liu, H. (1995), NeuroRule: A Connectionist Approach to Data Mining, *Proc. Intl. Conf. Very Large Databases* VLDB95.

Michie, D., Spiegelhalter, D.J., and Taylor, C.C. (eds) (1994), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood.

Mitchell, J. M. O. (1994), Classical Statistical Methods, *in* Michie (1994), 17-28.

Mitchell, T M. (1997), *Machine Learning*, McGraw-Hill.

Molina, R., Perez de al Blanca, N., Taylor, C.C. (1994), Modern Statistical Techniques, in Michie (1994), 29-49.

Punch, W.F., Goodman, E.D., Min Pei, Lai Chia-Shun, Hoyland, P., Enbody, R. (1993), Further Research on Feature Selection and Classification using Genetic Algorithms, in *Proceedings of the Fifth International Conference on Genetic Algorithms*, 557-564.

M.L. Raymer, W.F. Punch, E.D. Goodman and L.A. Kuhn, Genetic Programming for Improved Data Mining - Application to the Biochemistry of Protein Interactions, *Proc. Intl. Conf. Genetic Programming* GP-96, 375-380.

Parodi, A., Bonelli, P (1990), The Animat and the Physician, *Proc. First Intl. Conf.*

Riolo, R.L. (1987), Bucket Brigade performance: II. Default Hierarchies, *Proc. Second Intl.Conf. on Genetic Algorithms and their Applications*, 196-201.

Riolo, R.L. (1989), The emergence of default hierarchies in LCS, *Proc.Third Intl.Conf. on Genetic Algorithms and their Applications*, 322-337.

Rowher, R., Wynne-Jones, M., Wysotzki, F. (1991), Neural Networks, *in* Michie (1994), 84-106.

Saxon, S. (1998), Data Mining Techniques, *Tech. Rep.*, Computer Studies and Maths, UWE, UK.

Spears, W. M., DeJong, K.A. (1992), Using Genetic Algorithms for Supervised Concept Learning, in Bourbakis, N. G. (ed), *Artificial Intelligence Methods and Applications*, World Scientific.

Thrun, S.B., Bala, J., Bloedorn, E., Bratko, L., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fahlmann, S.E., Fisher, D.,Hamann, R., Kaufman, K., Keller, S., Kononenko, L., Kreuziger, J., Michalski, R.S. Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., Zhang, J. (1991), The Monk's Problems: A Performance Comparison of Different Learning Algorithms,*Technical Report: CMU-CS-91-197*, Carnegie Mellon University.

Wilson, S.W. (1983), Knowledge Growth in an artificial animal, *Proc. First Intl. Conf. On Genetic Algorithms and their Applications.*

Wilson, S.W. (1988), Bid Competition and Specificity Reconsidered, *Research Memo 54R*, Roland Institute.

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

Wilson, S.W. (1998), Generalization in the XCS classifier system, in *Proc. Third Annual Genetic Programming Conference* GP-98

Appendix

Coverage tables are mappings of the entire Monk's attribute space with each point marked as being correctly classified as 1 or 0 or incorrectly classified (X) and are closely based on those in Thrun (1991).

Holding	Jacket	Tie	Map to Column
Sword	red	yes	1
Sword	yellow	no	2
Sword	green	yes	3
Sword	blue	no	4
Sword	red	yes	5
Sword	yellow	no	6
Sword	green	yes	7
Sword	blue	no	8
Flag	red	yes	9
Flag	yellow	no	10
Flag	green	yes	11

Flag	blue	no	12
Flag	red	yes	13
Flag	yellow	no	14
Flag	green	yes	15
Flag	blue	no	16
Balloon	red	yes	17
Balloon	yellow	no	18
Balloon	green	yes	19
Balloon	blue	no	20
Balloon	red	yes	21
Balloon	yellow	no	22
Balloon	green	yes	23
Balloon	blue	no	24

Head	**Body**	**Smiling**	**Maps to Row**
round	round	yes	1
round	round	no	2
round	square	yes	3
round	square	no	4
round	octagon	yes	5
round	octagon	no	6
square	round	yes	7
square	round	no	8
square	square	yes	9
square	square	no	10
square	octagon	yes	11
square	octagon	no	12
octagon	round	yes	13
octagon	round	no	14
octagon	square	yes	15
octagon	square	no	16
octagon	octagon	yes	17
octagon	octagon	no	18

Monk's 1

These coverage tables are produced after a test iteration of XCS and indicate the population's ability to classify the every point in the attribute space.

The coverage table for perfect classification of the Monk' s 1 problem is shown in fig. A.1. This was produced in all but 5 of the 30 runs in the Monk's 1 experiment. Of the five other runs the coverage tables showed that the worst run had 21 misclassifications with the other four having no more than 8 misclassifications.

Monk's 2

This typical coverage table for the Monk's 2 problem indicates that although XCS has learnt the training data perfectly - see fig A.2 - the system classifies with ~90% accuracy over the whole set, indicating an that XCS has overfit to the training data.

Monk's 3

```
111111111111111111111111
111111111111111111111111
110000001100000011000000
110000001100000011000000
110000001100000011000000
110000001100000011000000
110000001100000011000000
110000001100000011000000
111111111111111111111111
111111111111111111111111
110000001100000011000000
110000001100000011000000
110000001100000011000000
110000001100000011000000
110000001100000011000000
110000001100000011000000
111111111111111111111111
111111111111111111111111
```

Figure A.1: The coverage table for 100% classification accuracy of the Monk's 1 problem as produced from 25/30 XCS

```
00000000000X0X0X000X0X0X
000X0X0X00X1010100X10101
000000000001010100010101
0X0X0X0X0110101001101010
000000000001010100010101
0X0X0101X110101001101010
000000000001010100010101
0X0101010110X0X00110X0X0
000101010110101001101010
011010101X0000001X000000
000101010110101001101010
011010101X0000001X000000
000000000001010100010101
0X0101010110X0X00110X0X0
000101010110101001101010
011010101X0000001X000000
000101010110101001101010
011010101X0000001X000000
```

Figure A.2: A typical coverage table for the Monk's 2 problem using the enumeration encoding.