CLASSIFIER SYSTEMS

Accuracy-based fitness allows similar performance to humans in static and dynamic classification environments.

Adrian R. Hartley

The University of Birmingham
School of Computer Science
Edgbaston
Birmingham, B15 2TT
United Kingdom

Email arh@cs.bham.ac.uk
Telephone (+44) (0)121 414 3711

Abstract

Traditionally within classifier systems the ability of a classifier to obtain reward (as measured by its strength) indicates the fitness of the classifier within the rule population. However Wilson (1995) proposed a new approach to fitness in terms of classifiers prediction accuracy. This paper presents experiments with two different classifier systems: Newboole (Bonelli et al. 1990) and XCS (Wilson 1995). Both systems demonstrate qualitative matches to data from perceptual category learning in humans. However, the different methods of fitness evaluation of classifiers alter the knowledge the systems learn and maintain. When fitness is based upon strength (Newboole) the system acquires knowledge to solve the classification problem, but when fitness is based on accuracy (XCS) the system acquires a more complete knowledge of the problem space. Further experiments show that the optimal covering map (Kovacs 1997) of knowledge that emerges in XCS allows the system to compensate rapidly in a dynamic classification environment, which is also more similar to human performance on a similar tasks.

1. Introduction

The ability to recognize quickly and accurately that which we encounter is fundamental to normal intelligent human behaviour. However how the learning of categories which objects in the world fit into takes place is still an unanswered question. But one thing is certain: much of the learning that does take place allows humans to cope with the changing environment they encounter. One of the most important aspects of human intelligence is its flexibility which has allowed humans to prosper in a dynamic world, humans do not suffer from the ills of old fashioned hard rule based artificial intelligence. Holland's (1986) answer to flexibility and escaping the brittleness of hard rule based systems was through the use of inductive classifier systems.

Classifier systems are soft rule based systems that interface and interact with their environment through messages. Large numbers of rules can be active simultaneously. All rules are of a condition/action form and are known as classifiers. The condition part specifies the set of messages for which the rule is active, and the action part specifies a message that is posted by an active rule. Messages are kept to standard lengths allowing conditions to be defined in the same standard length. This makes the generality of a condition simple to set with the use of #'s, which indicate a non-specified value as part of the condition and allow the condition part of a rule be satisfied by more than one environmental state. Generalization is essential for any learner, without it problems soon become intractable as the increase in size.

The rules maintain a value that reflects their usefulness within the system, these values are known as the rule's strength. A learning algorithm adjusts rule strengths and as the classifier system learns strength reflects a rule's usefulness over time. The difficult task is making the system positively inductive and not randomly inductive, this is to say that the system is more likely to generate useful new rules and not just any new rule. This task is carried out through the use of a genetic algorithm. This operates directly on selected rules, to produce new rules. New rules replace those rules that are known not to be useful to the system because they have low strength. However, the genetic fitness of a rule does not have to be defined by strength. It is possible to use a measure of the accuracy of a rule's reward prediction as the fitness of a rule for genetic competition. It is the competition between rules that gives an overall system a graceful way of handling conflicts and maintaining tentative hypotheses.

It has been proposed that classifier systems may be able to demonstrate similar learning to humans within a classification domain (Holland 1986, Riolo 1991, Sen 1996). Section 2 introduces some human data which has become a bench mark set of results within category learning. Section 3 presents the performance of Newboole, a classifier system with a strength based approach to fitness evaluation, on these tasks. This is a replication of Sen (1996). Section 4 presents the performance of XCS, a classifier system with a accuracy based approach to fitness evaluation, on these tasks. Section 5 discusses the difference in the knowledge the two systems acquire over the course of learning. In section 6 experiments are presented which demonstrate how accuracy based fitness allows fast recovery in a dynamic problem environment. The final section is a general discussion of these results and their implications.

2. The Shepard Tasks: Performance by humans

Categorization is considered basic to all our intellectual abilities (Estes 1994). Categorization "is the process of assigning objects (of whatever kind) to categories (which are collections of objects which are grouped together for some purpose)" (Lamberts 1997). The intelligent behaviour of perceptual categorization involves the classification of objects, translated from a physical representation in the external world to an internal representation by a perceptual system. However the ability to effectively acquire knowledge to classify the representations of objects from an agent's external world is no simple problem. Within human perceptual category learning a bench mark set of results have arisen from the early work of Shepard, Hovland, and Jenkins (1961). The tasks have a well defined difficulty ordering which increases as the amount of generalization within the problem decreases. Since generalization is so important within machine learning the Shepard tasks serve as a good starting point for a mechanistic implementation of category learning.

The 1961 study by Shepard et al. tested subjects on six basic types of classification problem. In each problem there were eight stimuli constructed from three binary dimensions. Four stimuli belonged to one category and the other four to a different category. These constraints result in six problem types, which are illustrated by the six cubes in figure 1. The vertices of the cubes represent individual stimuli. The circle vertices represent stimuli that belong to category A, and the square vertices represent stimuli that belong to category B. The faces of the cubes represent a constant value across one of the three dimensions that define the stimuli.

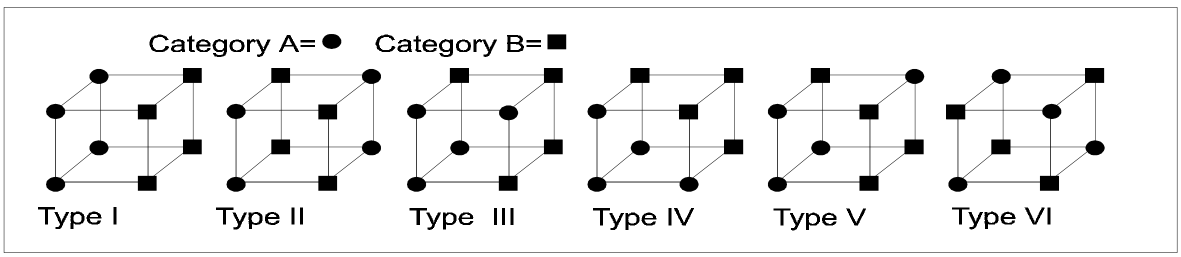

Figure 1: The six types of categorization problem tested by Shepard, Hovland, and Jenkins (1961). Diagram adapted from Nosofsky et al. (1994).

The dimension of the stimuli correspond to different physical properties, e.g. shape (square vs. triangle), colour (black vs. white), and size (large vs. small). Any assignment of four stimuli to each of the two categories can be reflected or rotated into one of the types illustrated in figure 1.

The simplest category structure is a Type I problem, where only information from a single dimension is relevant to solve the classification problem. The Type II problem has exactly two relevant dimensions and across these dimensions it is equivalent to an exclusive-or problem. The Type VI problem is the most complex, all three dimensions share equal importance in the solution to this categorization problem. This is equivalent to the 3-bit parity problem of machine learning, as there are no possible generalizations and the number of rules necessary to classify the stimuli is equivalent to the number of stimuli.

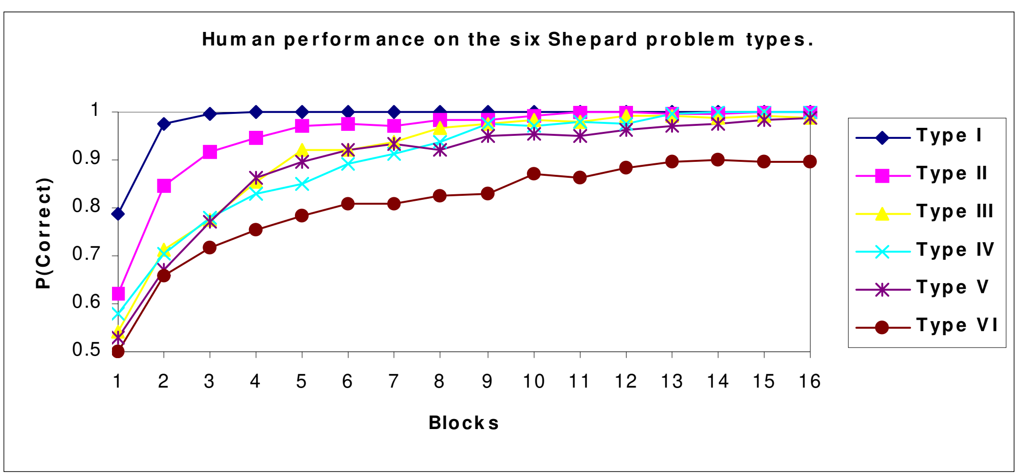

Figure 2 Average probabilities of correct classifications for each problem in each block of 16 trials, from Nosofsky et al. (1994). The probability of correct classifications was essentially 1 during later blocks for all problems except Type VI.

The other problems (Types III, IV, and V) are intermediate in their structural complexity, in between Type II and Type VI. All three dimensions are relevant but to different degrees. These categories have been describe as “single dimensions plus exception structures” (Nosofsky et al. 1994). Subtle differences between Types III, IV, and V are discussed by Nosofsky et al. (1994).

The Shepard et al. (1961) study was replicated by Nosofsky et al. (1994), who tested subjects using geometric forms. Stimuli varied along three binary dimensions and were fairly representative of the original Shepard et al. separable dimension stimuli. The logical structure of the six problems tested were as illustrated in figure 1. Subjects were tested on two types of classification problem to achieve a total of 40 subjects on each problem. The results from Nosofsky et al. (1994) replicate the findings by Shepard et al. (1961). Figure 2 shows Nosofsky et al. results as average probabilities of correct categorizations. These results show the robustness of the Shepard finding of a difficulty ordering of the six problem types.

3. Newboole

Newboole is a fast genetic based machine learning system. It is a supervised learning system, a simplification of the Boole classifier system (Wilson 1987). It runs in distinct cycles, making an educated guess biased by confidence as to which category inputs belong to. Every rule that matches the input is updated using a “symmetrical payoff penalty” scheme, which rewards correct rules and punishes incorrect ones within the same cycle. The “symmetrical payoff penalty” scheme allows Newboole to learn quickly as it severely penalizes incorrect rule so they quickly disappear. Like a standard Holland classifier this system has input and output interfaces and maintains a finite population of classifiers, but has no internal message list or memory.

Newboole’s classifiers consist of a single condition (taxon) and an action. For L-bit environmental messages, the taxon has a length L. The action is just a single bit, 0 or 1, representing the two possible category decisions from the Shepard tasks. The classifier population [P] is initialized by generating a random population of *P-max* classifiers (*P-max* is the maximum number of classifiers that can exist in the population [P]). This is implemented by the generation of random binary strings to the correct length of condition and action parts of classifiers, then each of the taxa in the taxon of classifiers have a probability (*P#*) of being flipped into a #. Initial strengths of classifiers are set to a common value (s_I).

Newboole performance component: The first component in the classifier run cycle receives an input string from the environment and returns its answer. This is done in two steps. First, the match set [M] is formed of all the classifiers whose taxa match the input string. After matching is performed a classifier is selected from [M], using a probability distribution over the strengths of the members of [M], and its action goes forward as the system output for the trial.

Newboole learning component: This follows the performance component. It is responsible for altering the classifiers’ strengths based on their performance. The set of correct classifiers [C] is formed. This is the subset of [M] that advocates the desired action for the current input. Also, the set of incorrect classifiers [Not-C] is formed ([Not-C] = [M] – [C]). A fraction e is deducted from the strengths of the members of [C] and a payoff quantity R is distributed among the members of [C], using a biased distribution so previously useful and or more general rules get a larger share (see Sen 1996, Hartley 1998). Finally, a large penalty fraction p is deducted from members of [Not-C] strengths.

Newboole uses the same genetic algorithm as Boole (see Wilson 1987, Hartley 1998) to evolve new rules from the existing rule base. Following the learning component, with a probability (*Prep)*, a number (O) of new offspring are produced to replace rules with low strengths. Parent classifiers are selected from [P], using a probability distribution over the strengths of all the classifiers. Genetic operators are then applied with a probability (*Pcross)* crossover is performed and changing each taxon allele with a probability (*Pmut*) implements mutation. Offspring are added to [P] and when this causes the maximum population size to be exceeded then classifiers are deleted with a probability proportional to the inverse of their strength. The discovery component has one further mechanism as well as the genetic algorithm called creation (Wilson, 1987). Creation only happens if during the performance cycle the match set [M] is empty. The taxon of the created classifier is a copy of the current input string to be classified, except alleles are flipped to #’s with a probability (*P#*). The action is chosen randomly and the strength of the classifier is set to the mean strength of [P].

The Shepard tasks as depicted in figure 1 are easily expressed in terms of binary strings for Newboole to learn. Newboole was run over the Shepard tasks consisting of 400 trials, with the following system parameters: *P-max* = 50, *P#* = 0.33, s_I = 100, O = 2, R = 1000, e = 0.114, p = 0.973, G = 0.667, *Prep* = 0.949, *Pcross* = 0.114, *Pmut* = 0.0118. The parameters *e, p, G, Prep, Pcross*, and *Pmut* where optimized using a meta genetic algorithm to avoid any biasing of the system. Optimization of system parameters was done to minimize the error across all six problem types.

The training exemplars for each problem were presented to the system in randomized blocks. A randomized block contained all the exemplars of the problem in a random order. Performance over each block was recorded, so the performance measure on a block reflects the percentage correct over the previous eight trials. The results obtained from experiments on the six Shepard type problems are presented in figure 3. Each data point was averaged over forty runs of Newboole. The performance of Newboole on the Shepard tasks qualitatively matches that of human subjects (see figure 2), in that they show the same difficulty ordering.

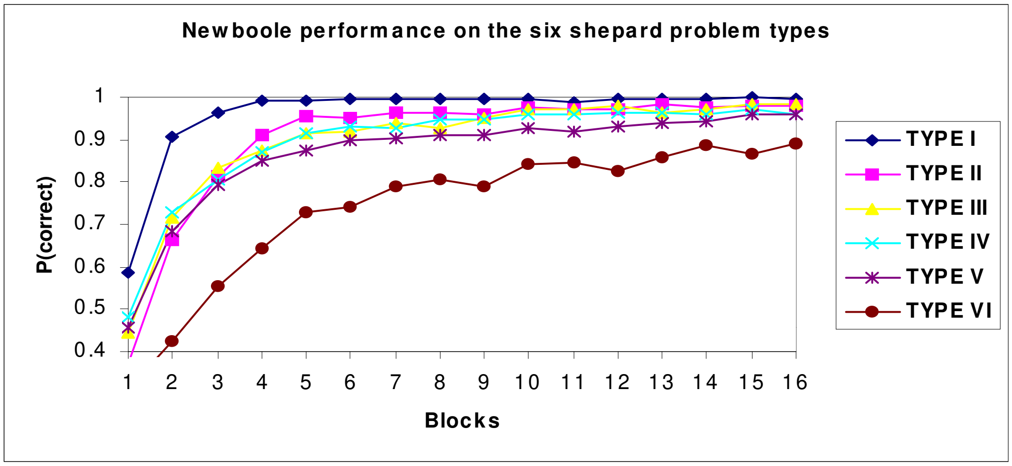

Figure 3 Average probabilities of correct classifications for each problem in each block of 8 trials. The probability of correct classifications was essentially 1 during later blocks for all problems except Type VI.

4. XCS

XCS is a more complex classifier system in the rules it maintains and its operation. The strength or predicted payoff of classifiers is only relevant in the selection of actions, thus classifiers are free to map any area of the input/action/payoff space. As a result XCS tends to form "complete and accurate mappings X x A $\Rightarrow$ P from inputs and actions to payoff predictions … which can make payoff-maximizing action-selection straightforward" (Wilson, 1995). Kovacs (1996) argues that "by relieving payoff prediction of its double role Wilson has significantly improved the classifier system architecture".

XCS like other classifier systems has a performance, learning, and discovery component. However, it also has others not found in standard classifier systems, e.g. it uses a MAM ("moyenne adaptive modifiee", Venturini 1994) technique in the updating of its parameters, it employs a niche genetic algorithm, and it makes use of subsumption deletion. Another difference is that XCS makes use of macroclassifiers, these are just a type of classifier with a numerosity parameter. Therefore when new classifiers are introduced XCS checks to see if there is an existing classifier with the same condition and action, if there is its numerosity is incremented by one. The same happens in classifier deletion: numerosity is decreased by one and a classifier is only actually deleted if it has a numerosity of zero. If either a chosen or random initial population is used the parameters are set to constant default values.

XCS performance component: like Newboole XCS has no internal message list and strings from the input interface are sent directly to the module that generates the set of matching classifiers. The system receives an input string and returns its answer by first forming a match set [M] from classifiers in the population [P]. Predictions (pa_i) for each action (a_i) are computed using a fitness weighted average and put in an array of predictions. Advocacy thresholds are use to reduce the effect of relatively untested classifiers on system predictions. An action is selected, the action set [A] of classifiers in [M] that advocate the selected action is formed and the action is translated into the system output.

XCS learning component: A Q-learning like procedure is implemented to update classifier parameters (see Wilson 1995, Hartley 1998). Learning in XCS is different for single step and multi step problems. For single step problems the predictions pa_i of classifiers in [A] are updated towards the current cycle's reward. For multi step problems, the previous cycles action set $[A]_{-1}$is updated using the sum of the previous cycle's reward and the maximum of the prediction array discounted by $\gamma\{P \leftarrow \textit{previous cycle's reward} + (\gamma \times max(pa_i))\}$ or as the single step update if the problem is completed in a single step. The multi step version of XCS is not relevant for the Shepard tasks. Both Wilson (1995) and Kovacs (1996) update the parameters in the classifiers belonging to [A] or $[A]_{-1}$ using the same sequence, which is adopted in this implementation.

The genetic algorithm in XCS operates on the action set (Wilson 1998) and always generates two offspring. When the number of time steps since the genetic algorithm was last called for a particular [A] exceeds the GA threshold (θ) then new classifiers are introduced to [P]. 2 Parent classifiers are selected from [A] probabilistically over their fitnesses. Genetic operators are applied to the copies of the parent classifiers. Crossover occurs with a probability (χ) per pair of chromosomes (bitstrings) and then mutation of each allele (bit) occurs with a probability (μ) per allele. Offspring are checked against the classifiers in [P] to see if an existing classifier subsumes them. New classifiers are subsumed if the existing classifier: i) has experience exceeding a subsumption threshold value ii) if it has an accuracy of 1 and iii) if it logically subsumes the new one (the inputs it matches are a superset of the inputs the new classifier matches). If a new classifier is subsumed it is discarded and the subsuming classifier's numerosity is incremented. If the population size M exceeds the limit N, classifiers are deleted to return M to N. Classifiers are deleted probabilistically to maintain the allocation of system resources across genetic niches. XCS can also introduce classifiers to [P] through a similar method to creation in Newboole known as covering. Covering happens when [M] is empty to ensure there is always a matching classifier and is implemented in the same manner as creation in Newboole.

The architecture of XCS led Wilson (1995) to originally propose the generalization hypothesis and later Kovacs (1996) to extend the generalization hypothesis with the optimality hypothesis. The generalization hypothesis states that due to the interaction between accuracy based fitness and a niche based genetic algorithm evolutionary pressure is directed towards evolving classifiers that are both accurate and general. The optimality hypothesis continues this stating that, an XCS system should eventually evolve the maximally general classifiers when it is able to sample sufficiently (see Kovacs 1996).

Like all reinforcement learning systems XCS faces the exploration/exploitation dilemma where the system must decide between exploration which maximizes learning potential and exploitation which maximizes performance. Typically more resources are allocated to exploration early in problem solving and then as a system learns about its environment more resources can be shifted to exploiting the learning. The dilemma is solved in XCS experiments by performing first, an exploit, then an explore, cycle for each input. On exploit cycles, action selection from the prediction array was deterministic, but learning and discovery components were disabled. On explore cycles the system's actions were selected at random from those advocated in the prediction array, and learning and discovery components operated normally. Results are only recorded for exploit cycles.

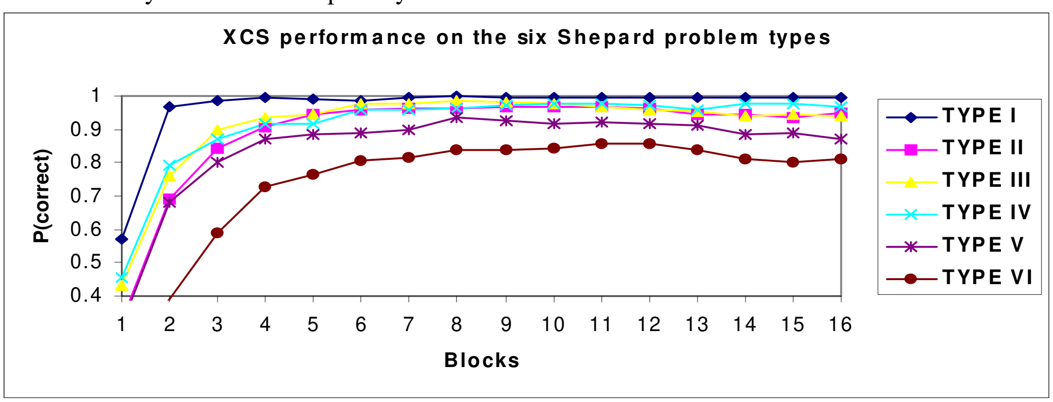

Figure 4 Average probabilities of correct classifications for each problem in each block of 8 trials. The probability of correct classifications was essentially 1 during later blocks for all problems except Type V and VI.

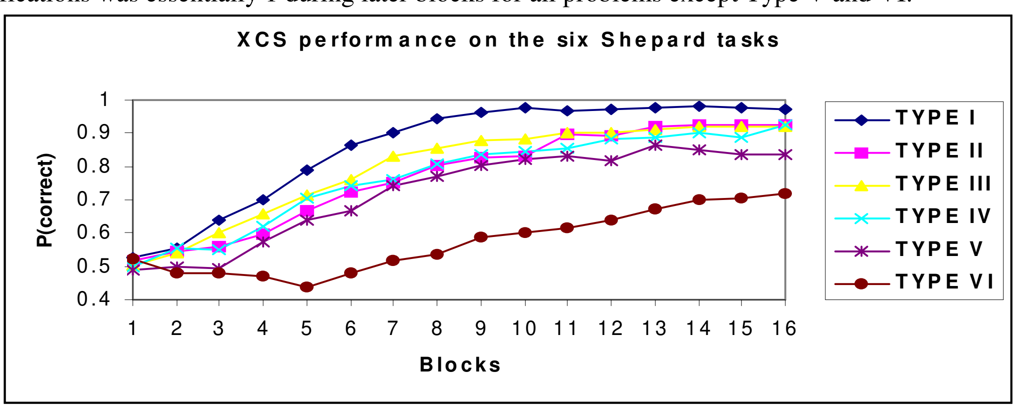

Figure 5 Average probabilities of correct classifications for each problem in each block of 8 trials. The probability of correct classifications rose essentially to 1 by the 30th block for Type II, the 38th for Type III and IV. Type V had a probability of 0.93 at the 50th block and the Type VI a probability of 0.86.

The representations of the Shepard tasks from Newboole are used for XCS. The experiments were run consisting of 400 exploit trials, with the following system parameters: Initial population size = 50, Maximum population size (N) = 50, Classifier deletion procedure is deletion probability b, Learning rate (β) = 0.2, Accuracy criterion (ε_0) = 0.01, Accuracy falloff rate (α) = 0.1, Delta (δ) = 0.1, Advocacy threshold = 31, GA threshold (θ) = 29, Subsumption threshold = 8, Crossover rate (χ) = 0.890, Mutation rate (μ) = 0.0902, Hash probability ($P\#$) = 0.33. The parameters β, advocacy threshold, θ, subsumption threshold, χ, μ, were optimized using a meta genetic algorithm, as in the Newboole experiment, to avoid any biasing of the system. Optimization of system parameters was done to minimize the error across all six problem types. The same training methods and performance recording methods as used in the Newboole experiment were used for XCS. The performance of XCS on the Shepard tasks (figure 4) qualitatively matches that of human subjects in that they show the same difficulty ordering, though not as well as Newboole.

The experiment depicted in figure 4 was rerun, but this time XCS was given no initial population as with previous work with XCS (Wilson 1995, 1998, Kovacs, 1996). The experiments were run consisting of 400 exploit trials, with the following system parameters: Initial population size = 0, Maximum population size (N) is 50, Classifier deletion procedure is deletion probability b, Learning rate (β) = 0.267, Accuracy criterion (ε_0) = 0.01, Accuracy falloff rate (α) = 0.1, Delta (δ) = 0.1, Advocacy threshold = 1, GA threshold (θ) = 13, Subsumption threshold = 18, Crossover rate (χ) = 0.706, Mutation rate (μ) = 0.2, Hash probability ($P\#$) = 0.33. The parameters β, advocacy threshold, θ, subsumption threshold, χ, μ, were optimized again. The results obtained from experiments rerun on the six Shepard type problems are presented in figure 5.

The performance of XCS on the Shepard tasks qualitatively matches that of human subjects as well as Newboole when no initial population is used. However, the speed of the problem solving is slower in XCS when no initial population is used compared to when an initial population is used (Newboole, figure 3; XCS, figure 4). This is to be expected as the system is starting with no knowledge. Newboole and XCS are equivalent on this problem even though Newboole is nominally a supervised learner and XCS a reinforcement learner, because there are only two actions and both systems immediately cause a difference between the correct and incorrect matching classifiers. When XCS starts with no initial population learning for the Type II problem reaches maximum performance significantly before the more complex problem types.

5. XCS and Newboole learn different representations for the Shepard tasks.

Newboole and XCS both acquire a rule base to solve the Shepard type categorization tasks. Also, similarly both systems demonstrate a qualitative match to the performance in humans. Sen (1996) comments about the structure of rules needed to solve the Shepard type categorization tasks. For the simpler problems, the rules to be learnt are more similar to each other in both the position and the numbers of #'s values. In the Type II problem the number of #'s in rules and their positions is constant, whereas in the Type III problem the number of #'s is constant but the position of the #'s within the rules changes. The difficulty ordering of the learning curves in the tasks can be thought of as reflecting the size of the optimal rule sets (Kovacs, 1997) for each problem. The optimal rule set, is the smallest set of the most general non-overlapping rules that are always correct for a specific problem. As the Shepard types increase in difficulty so does the number of rules in the optimal rule set.

In Newboole due to strength based fitness evolutionary pressure is directed solely to the development of the optimally general rule set. Newboole acquires only knowledge about the correct category decision to make for each item in the categorization problem. Incorrect classifiers are punished heavily so these have a low chance of survival and this knowledge is lost quickly. Newboole tends to learn the most rewarding action for each environmental input (called a best action only map) and has no knowledge about what it is like to take other actions.

However, XCS does more than just develop the best action only map set for each of the Shepard tasks. Within XCS evolutionary pressure tends towards the development of a complete covering map of knowledge in terms of optimal rule sets, due to accuracy based fitness. XCS maintains the most general rules that always correct (00# / 0, 11# / 0, 01# / 1, 10# / 1, for the Type II task) and also the most general rules that are always wrong (00# / 1, 11# / 1, 01# / 0, 10# / 0, for the Type II task) for these are the most accurate rules (Kovacs 1996, Hartley 1998). The complete covering map of knowledge emerges quickest to dominate the population when covering is used to introduce classifiers to the population because covering introduces fewer over and over general rules than an initial random population. XCS learns not only the optimal category decision for each item in the categorization task, but also the effects of making sub-optimal decisions for items.

6. A dynamic problem environments

It is known that humans have the ability to rapidly compensate for changes within an environment that they have experienced. Human subjects are able to learn artificial category structures such as those in the Shepard tasks, but they are also able to rapidly adjust the category structure they have learnt to cope with some systematic change. Humans can completely relearn a categorization very quickly as long as the new category structure arises from a systematic change in a previously learnt category structure. In terms of the Shepard tasks, if during the learning of a problem when performance has been high for some time all the members of one category are switched to the other category and vice versa, then this will disrupt categorization performance only briefly in humans.

This ability of fast recovery was tested within Newboole and XCS, using the same parameters as above. Two experiments were run one using the Type I problem the other using the Type V problem. The training exemplars for each problem were presented to the systems in randomized blocks. After the first 200 trials the categories were switched so all members of category A became members of category B and visa versa. Performance was measured in the same manner as in the first set of experiments. The results obtained from experiments on the Type I problem are presented in figure 6 and the results from the Type V problem are presented in figure 7. Each data point was averaged over forty runs.

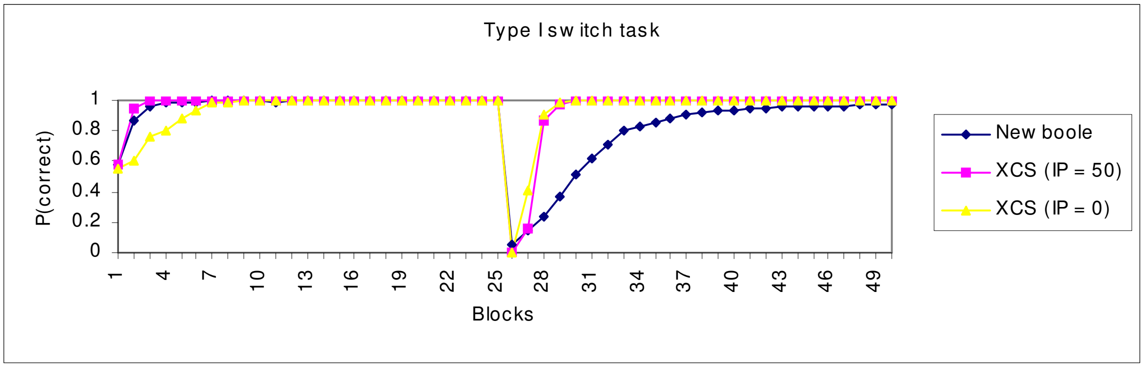

Figure 6 Average probabilities of correct classifications for a Type I problems over 400 trials with a category switch on the 200th trial.

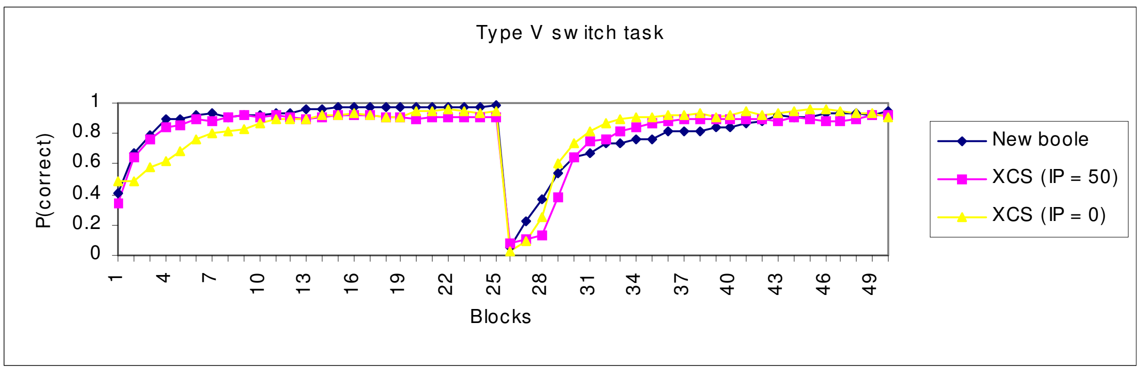

Figure 7 Average probabilities of correct classifications for a Type V problems over 400 trials with a category switch on the 200th trial.

The results show that in both cases for Newboole the learning of the categorization is slower after the switch than it was to acquire the original category structure. Newboole is also slower to recover from the change in the problem environment than XCS. With XCS recovery is fastest when the system is started with no initial population for both problem types. However, when XCS starts with no rule base acquisition of the initial problem structure is slower.

Newboole is slower to recover because its rule base specializes very quickly and removes rules that do not obtain high strengths. As no rules, which advocate incorrect actions for environmental inputs are able to survive when there is a change in the problem environment Newboole suddenly has a completely incorrect rule base and has to evolve the new rules needed because of its previous overspecialization.

XCS maintains a broader knowledge of the problem space and it is the evolution of the complete covering map of knowledge that allows the system to recover faster. Prior to the switch XCS learns rule describing both the desired action and undesired action for each exemplar. After the switch if the optimal covering map of knowledge is complete the system has all the rules it needs to solve the problem - it just needs to alter the reward each rule predict for the performance to be optimal again. In the type I problem the optimal covering map is very simple (only 4 rules) and exists completely before the switch so recovery is very fast. But, in the type V problem the optimal rule set is larger (12 rules) and the slower recovery in figure 7 may be due to the incompleteness of the covering map of knowledge at the point of switch. The ability of XCS to evolve a complete covering map of knowledge gives it an advantage in coping with a dynamic problem environment. This observed advantage is related to the advantages Kovacs (1999) hypothesized for complete covering maps when exploration is difficult, e.g. in slowly changing environments and multi step environments.

7. General discussion

The results from the Shepard tasks suggest that the learning mechanism within XCS and Newboole can adapt to the different amount of generality within the Shepard type problems capturing the difficulty ordering that has been attributed to an attentional mechanism in humans. Newboole learns about rewarding decisions whereas XCS learns not only to solve problems but also about the problem space forming a more complete knowledge base to aid its decisions. It is interesting to note that the learning behaviour displayed in this paper is inherent to the systems, as the systems were not optimized to replicate human performance but simply to maximize performance across all the Shepard type problems. In fact if during optimization if the probability of mutation is not limited it increases to change the operation of the genetic algorithm into random string generation. Random string generation gives better performance on these small problems but the difficulty separation disappears. Essentially the system architecture is altered as the operation of the discovery component changes, because no longer do the strongest or most accurate rules influence the new rules introduced.

In both supervised or reinforcement learning schemes within classifier systems when fitness is based on strength the incorrect or low reward receiving classifiers will tend be quickly replaced. However, Kovacs (1999) proposes a method of maintaining complete covering maps in strength based classifier systems called weeding. Weeding effects the fast removal of over general classifiers, which may allow the survival of lower reward obtaining classifiers. Complete covering maps would still seem difficult to maintain over a long period of learning without accuracy based fitness. Accuracy based fitness needs more memory resources, but benefits are demonstrated by the ability of XCS to develop complete covering maps of rules that allow rapid recovery in a dynamic problem environment.

Psychologists have used standard neural networks more frequently to model human learning data. However, although standard neural networks are able to learn categorization tasks they do not have the ability to refine their representations of solutions. XCS refines its representations of solutions towards the simplest possible in terms of the complete covering map. The refined representations of XCS give it a robustness in the dynamic categorization environment where a neural network would suffer from catastrophic interference.

This paper shows how two different classifier systems are both able to demonstrate similar learning performance over a set of classification tasks when compared to humans. However, it is only XCS with its accuracy based approach to fitness that is able to demonstrate rapid recovery in a dynamic problem environment due to its acquisition of a complete covering map of knowledge. The learning of complete covering maps of knowledge cost more in terms of memory and learning resources, but has definite advantages in classification environments that may have some aspect of change.

References

- Bonelli, P., Parodi, A., Sen, S., & Wilson, S. W. (1990). NEWBOOLE: A fast GBML system. In the *Proceedings of the Seventh international conference on machine learning*. 153-159. San Mateo, CA: Morgan Kaufman.
- Estes, W.K. (1994). *Classification and cognition.* New York: Oxford University Press
- Hartley, A. R. (1998). Genetics based machine learning as a model of perceptual category learning in humans. MSc Thesis. Computer Science Department, University of Birmingham, UK.
- Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (eds), *Machine learning: an artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.
- Kovacs, T. (1996). *Evolving optimal populations with XCS classifier systems*. MSc Thesis. Computer Science Department, University of Birmingham, UK.
- Kovacs, T. (1997). XCS classifier system reliably evolves accurate, complete and minimal representations for Boolean functions. In the *Second On-line World Conference on Soft Computing in Engineering and Design and Manufacturing*, http://www.cs.bham.ac.uk/~tyk/online97/online97.html
- Kovacs, T. (1999). Strength or Accuracy ? (In preparation).
- Lamberts K. (1997). Process models of catagorisation. In K. Lamberts & D. Shanks (Eds.), *Knowledge, concepts and catagories.* London: UCL Press.
- Nosofsky R. M., Gluck, M. A., Palmeri, T. J., McKinley, S. C. & Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland and Jenkins (1961).
- Riolo R. L. (1991). Modeling simple human category learning with a classifier system. In Proceedings of the 4th International conference on genetic Algorithms, 324-333. San Mateo, CA: Morgan Kaufman.
- Sen S. (1996). Modeling human categorization by a simple classifier system.
- Shepard, R. N., Hovland, C. I., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs,* 75 (13 Whole No. 517).
- Shepard, R. N., Hovland, C. I., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs,* 75 (13 Whole No. 517).
- Venturini, G. (1994). Apprentissage Adaptatif et Apprentissage Supervise par Algorithme Genetique. These de Docteur en Science (Informatique), Universite de Paris-Sud.
- Wilson S. W. (1987). Classifier systems and the animate problem. *Machine Learning*. 2, 199-228.
- 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 the Proceedings of the Third Annual Conference on Genetic Programming. J. Koza et al. (eds) San Francisco, CA: Morgan Kaufmann.