

# On Generalisation in Michigan-Style Fuzzy Classifier Systems

Brian Carse and Anthony G. Pipe

UWE Learning Classifier Systems Group Technical Report – *UWELCS02-006*

Faculty of Computing, Engineering and Mathematical Sciences  
University of the West of England  
Coldharbour Lane  
Bristol BS16 1QY, UK  
Brian.Carse, Anthony.Pipe@uwe.ac.uk

**Abstract.** The ability of a rule-based system to represent generalisations is of great importance. Generalised rules allow more compact rule bases, scalability to higher dimensional spaces, faster inference and better linguistic interpretability. The issue of rule generalisation, and the interplay between general and specific rules in the same evolving population, has received a great deal of attention in the discrete-valued classifier system research community. The same issue does not appear to have received a similar level of attention in the case of fuzzy classifier systems. While it is true that generalised rule representations have been incorporated in a number of reported fuzzy classifier systems, often as an aside to other significant issues, this important feature has not yet been concentrated on in detail. The intention of this contribution is to raise awareness of the issue of generalisation in the fuzzy classifier system so that the issue may be brought under closer scrutiny. Early experimental results using a test-bed specifically designed to test the ability of the fuzzy classifier system to learn an optimal collection of co-existing general and specific fuzzy classifiers are described and discussed.

## 1 Introduction

The fuzzy classifier system is a machine learning system which employs linguistic rules and fuzzy sets in its representation and an evolutionary algorithm for rule discovery. It therefore combines an easily understood representation (as opposed to, for example, neural network approaches) with a general purpose search method. In order to exploit the fuzzy representation to the full, the ability to learn generalisations is of great importance. Generalised rules allow more compact rule bases, scalability to higher dimensional spaces, faster inference and better linguistic interpretability. The issue of rule generalization, and the interplay between general and specific rules in the

same evolving population, has received a great deal of attention in the discrete-valued classifier system research community (e.g. [10, 21, 22]). The same issue does not appear to have received a similar level of attention in the case of fuzzy classifier systems.

In a discrete-valued classifier system, classifiers (rules) are represented as bit strings and generalised rules are obtained by using '#' symbols ("don't cares") in the classifier syntax. The '#' symbol matches both '0' and '1' so, for example, the classifier condition (rule antecedent) 11## matches the input messages 1100, 1101, 1110 and 1111. However, such a generalisation capability has been known for a long time to provide problems for discrete valued classifier systems in terms of learning an optimal mix of general and specific rules, and this also applies to the fuzzy case. A major problem is that of proliferation of "overgeneral" rules: rules which match many input states but whose outputs are only correct for a subset of input states and are incorrect for others. Despite being unreliable, such overgeneral rules can have more influence and better chances of survival (under action of the evolutionary algorithm) than other more specific and correct rules with which they compete.

Traditional Michigan-style classifier systems [20] have been "strength-based" in the sense that a classifier accrues strength during interaction with the environment (through rewards and/or penalties). This strength is then used for two purposes: resolving conflicts between simultaneously matched classifiers during learning episodes; and as the basis of fitness for the evolutionary algorithm. A number of problems arise from this dual use of classifier strength (particularly if a pan-mictic GA is applied). These include:

1. The cooperation/competition problem. High-strength, potentially cooperative classifiers go on to compete under the action of the evolutionary algorithm.
2. Over-general rules with relatively high (but inconsistent) payoff can come to dominate the population.
3. In some environmental states, the maximum payoff achievable (by performing the best possible action for that state) may be relatively low. Although a classifier might be the best that can exist for that state, it can be eradicated from the population by other classifiers that achieve higher rewards in other states. This results in gaps in the system's "covering map".

In [10] a completely different approach is taken in which a classifier's fitness, from the point of view of the evolutionary algorithm, is based on its "accuracy" i.e. how well a classifier predicts payoff whenever it fires. Such an accuracy based approach offers a number of advantages. Firstly, it can distinguish between accurate and overgeneral classifiers: an overgeneral classifier will have relatively low accuracy since payoff will vary according to the input states covered by the classifier. Indeed, it has been shown that the accuracy-based approach can lead to evolution of optimally general classifiers. Additionally it can maintain both consistently correct and consistently incorrect classifiers which allows learning of a complete "covering map". A potential drawback of the accuracy-based approach is that it is likely to require larger populations of classifiers.

The ability of a *fuzzy* rule-based system to incorporate generalisation is largely a matter of fuzzy set and fuzzy rule representations (although, of course, successful *learning* of generalization also depends on operators and algorithms used). Indeed, it

could be argued that fuzzy representations permit a more rich collection of ways in which generalisation can be represented compared to the discrete-valued counterpart. Possible ways in which generalisation might be represented in fuzzy classifier systems include:

#### Generalisation via Fuzzy Set Membership Function Representation

- Allowing fuzzy set membership functions to be learned by the GA (the approach employed in [1]). By evolving membership function centres and widths, large areas in the input space which require similar outputs can be identified by the GA with a subsequent reduction in the rule-base size. This approach is sometimes called an “approximate Mamdani fuzzy system”, and has the potential drawback that evolved rules may not have an easily understood meaning in terms of their linguistic interpretability.
- Simultaneous fuzzy set partitions at different levels of granularity. For example, a fuzzy system could use sets {NL,NS,Z,PS,PL} and {N,Z,P} interchangeably according to the granularity required in different regions of the input space [2, 11, 12, 13].

#### Generalisation via Fuzzy Rule Representation:

- Allowing ORs and NOTs in the rule syntax. Possible rule antecedents might then be of the kind:
  - if (x1= {NL OR NS}) THEN...; sometimes called a “disjunctive normal form (DNF) Mamdani fuzzy system” (e.g. [5, 14])
  - if (x1=NL) OR (x2=PS) THEN... (e.g. [15])
  - if NOT(x1=PL) THEN...
- Allowing “don’t care” markers in the rule syntax; for example a system with three input variables x1, x2 and x3 might have rules such as
  - IF (x1=NL) AND (x2=PL) AND (x3 = DON’T CARE) THEN...

The paper is organised as follows. Section 2 outlines related work. Section 3 describes results of initial experiments in evolving an optimal collection of simultaneously present general and specific fuzzy rules using a test problem specifically designed to test the ability of the fuzzy classifier system to learn such an optimal collection of rules. Section 4 concludes.

## 2. Related Work

This section provides an overview of related work on Michigan-style fuzzy classifier systems; that is, systems which use evolutionary algorithms for classifier discovery and where the unit for evolutionary operations and credit assignment is the individual classifier. For reasons of space, it is not possible to provide an overview of Pittsburgh classifier systems, where the unit for evolutionary operations and credit assignment is a complete rule-base; the interested reader can find further details in a tutorial paper [3] and in the encyclopaedic text Genetic Fuzzy Systems [4].

The first description of a Michigan-style fuzzy classifier system is given by Valenzuela-Rendón [5]. Closely modeled on the discrete-valued Holland-style classifier

system [20] this system contains a fixed size rule-base of fuzzy classifiers and a fuzzy message list. An individual rule is represented as a binary string that encodes the membership functions of the fuzzy sets involved in the rule. To illustrate the representation used, consider a variable  $X$  for which there are 4 component fuzzy sets ( $F_1, F_2, F_3, F_4$ ). The string 1001 represents the fuzzy set which is the union of  $F_1$  and  $F_4$ . The representation is therefore capable of supporting two types of general rules: rules with conditions of the form IF ( $X$  is SMALL) OR ( $X$  is LARGE), and rules which contain "don't cares" for one or more input variables. In this system, the required length of the string is equal to the number of fuzzy sets selected for that variable. Each string is preceded by a non-fuzzy binary tag indicating the variable to which that fuzzy set refers. Fuzzification at the input and defuzzification at the output are accomplished using the concept of "minimal messages". The input unit generates a set of minimal messages with activity levels equal to the degree of membership of the input variable to the fuzzy sets which the minimal messages represent. The satisfaction level of a classifier's condition is equal to the maximum activity level of the minimal messages which it matches. The activity level of the whole classifier is equal to the minimum of the satisfaction levels of all its conditions (fuzzy intersection). Each matched classifier then fires, posting its action as a message (together with its activity level) to the message list. Defuzzification is performed by decomposing each output message into its corresponding minimal messages, adding up the activity levels of each minimal message for each output variable, then performing a fuzzy union over the component sets represented by the minimal messages multiplied by their activity levels. Finally the centre of gravity of the union is calculated, and assigned to the output variable. Credit is allocated to individual classifiers according to how closely each classifier predicts the correct output. The payoff distribution scheme is therefore, as Valenzuela-Rendón states, not pure reinforcement learning. True reinforcement learning is introduced to the system by Valenzuela-Rendón in [6].

Parodi and Bonelli [7] present a fuzzy classifier system which automatically learns fuzzy relations, fuzzy membership functions and classifier weights. Since credit assignment and genetic operations are carried out at the level of the individual rule, we characterise this system as Michigan-style. Parodi and Bonelli employ a real number representation for rules, with each rule encoding centres and widths of triangular fuzzy set membership functions for each input and output variable. The GA crossover operator exchanges randomly selected fuzzy sets between pairs of rules. Mutation uses real number "creep" which modifies the centres and widths of membership functions within a rule. The classifier population consists of a fixed size list of such rules. Each rule has associated with it an "output weight" which is set equal to the strength (fitness) of that rule. The rule strength therefore performs a dual function: firstly it forms the basis of selection and replacement for the GA and secondly it allows stronger rules to take a bigger part in decision making than weaker ones. In this system, "don't care" symbols are not allowed and the capabilities for generalisation of the classifier system are via learning appropriate widths of triangular membership functions.

In [8], Ishibuchi, Nakashima and Murata apply a Michigan-style fuzzy classifier to the problem of pattern classification. A fixed sized classifier population is used, and

fuzzy set membership functions are set in advance. Rule antecedents are encoded as a string of fuzzy set labels, one for each input variable. “Don’t care” symbols are allowed. This antecedent representation is similar to that used by Bonarini (described below), and that used in experiments described later in this contribution. The rule consequents are an output class (for the classification problem considered) and a certainty factor. These latter are derived using a heuristic procedure prior to fitness evaluation, and the GA operates on the rule antecedent only. Credit assignment methods are incorporated which explicitly address the classification problem. In this system, classifier generalisation capability is therefore provided by the use of “don’t cares” in the rule syntax.

As far as we are aware, all of the fuzzy classifier systems described above employ a pan-mictic GA, i.e. one which operates selection and replacement operators at the level of the whole population. This raises the question of the cooperation/competition problem in learning classifier systems: classifiers which “cooperate” in achieving desired behaviour (perhaps even more important in fuzzy classifier systems compared to their discrete-valued counterparts) later go on to “compete” under the actions of the evolutionary algorithm.

This problem is directly addressed in the case of fuzzy classifier systems in Bonarini’s work (e.g. [3], [9] which proposes ways to deal with the problem in a Michigan-style fuzzy classifier system called ELF. In ELF, the GA operates on sub-populations of fuzzy classifiers with the same antecedent but different consequent (somewhat akin to the way that the GA is applied in Wilson’s discrete-valued classifier system XCS [10]). The fuzzy classifier representation used in ELF is similar to that used in several other studies – rule antecedents and consequents are integer strings representing fuzzy set labels for each input and output variable respectively. Generalisation in this representation is, again, achieved by allowing “don’t cares” in rule antecedents, although, as far as we are aware, ELF has not been evaluated using generalised rules in the literature. Bonarini’s fuzzy classifier system has been applied to a wide range of learning tasks, with extensive evaluation of different reinforcement learning algorithms [3].

In [14], González and Pérez introduce a genetics-based fuzzy learning system called SLAVE (Structured Learning Algorithm in Vague Environments). The SLAVE system has since been continuously improved (e.g. in [15, 17]). Although not strictly a classifier system, SLAVE does use the genetic algorithm to evolve co-adapted collections of fuzzy rules, and therefore is highly germane to the subject of the current contribution. In this section, we shall only concentrate of the features of SLAVE which are concerned with the representation and learning of generalisation; the SLAVE system is a sophisticated system with many novel features and there is not enough space to give full details in the current contribution. The fuzzy rule representation employed in SLAVE encodes fuzzy rule antecedents as bit strings where each bit encodes the presence or otherwise of a fuzzy set in the rule for each input variable. For example, for a system with three inputs  $x_1$ ,  $x_2$ ,  $x_3$ , each of which is covered by three fuzzy sets (N, Z, P), the string 110001111 represents the rule

IF ( $x_1 = (N \text{ OR } Z)$ ) AND ( $x_2 = P$ ) AND ( $x_3 = (N \text{ OR } Z \text{ OR } P)$ ) THEN...

Note that, in this case, the rule essentially contains a “don’t care” for input variable  $x_3$  so the antecedent can be rewritten as  $(x_1 = (N \text{ OR } Z)) \text{ AND } (x_2 = P)$ . In a later version, SLAVE-2 [17], the representation is further extended to encode general rules. In SLAVE-2, each rule antecedent is encoded as two bit strings. The first is called the VAR (variable) string and its length is equal to the number of input variables. A ‘1’ indicates a particular variable is active in the rule, a ‘0’ indicates the variable is inactive. The second string is the VAL (value) string which employs an encoding scheme the same as in the original SLAVE. For example, assuming the same three variables and fuzzy sets used above, the string

(VAR 011 VAL 110001111) represents the antecedent

IF ( $x_1 = \text{‘Don’t Care’}$ ) AND ( $x_2 = P$ ) AND ( $x_3 = (N \text{ OR } Z \text{ OR } P)$ ) THEN...

This antecedent can be rewritten as the highly general ( $x_2 = P$ ), where variable  $x_1$  is eliminated at the variable level, and  $x_3$  is eliminated at the value level. In addition to this powerful rule representation, SLAVE also incorporates additional genetic operators specifically designed to provide exploration in the space of general/specific rules. These operators (called OR, AND and Generalisation) are combined with crossover to create more general antecedents (in the case of OR and Generalisation) or more specific antecedents (in the case of AND) in offspring.

In [18, 19], Hoffmann and Pfister apply a “messy GA”[23] to generation of fuzzy rule bases. In this work, each rule antecedent is encoded as a variable length list of integer pairs (*variable, label*) that refer to a particular fuzzy set for a particular variable. For example, for the same three input example described above the fuzzy sets (N, Z, P) might be encoded as (1, 2, 3) respectively. The sequence (1,2), (2,3), (3,1) would then encode the antecedent

IF ( $x_1 = Z$ ) AND ( $x_2 = P$ ) AND ( $x_3 = N$ ) THEN...

Generalisation using this representation occurs when an integer pair for a particular input variable appears more than once in the encoding. For example, the sequence (1,2), (2,3), (3,1), (1,1), (1,3) effectively encodes the antecedent ( $x_2 = P$ ) AND ( $x_3 = N$ ) and variable  $x_1$  does not contribute to the rule. In this way, the representation can encode both highly specific and highly general rules as required.

### **3. Experimental Investigation**

#### **3.1 The Test Problem**

The test problem chosen is that of learning a fixed fuzzy rule base which has been designed to incorporate an optimal mix of specific and general rules. Although this may appear somewhat artificial, the test problem has been designed in the same spirit as the multiplexor problem which is extensively used in evaluation of discrete valued classifier systems. For both problems, high-performance and economical solutions,

which contain an optimal combination of general and specific rules, are known in advance and this allows us to evaluate, under relatively controlled conditions, the performance of the learning classifier system. The intention here is that investigation of such a problem will give us some confidence in how well the learning approach will extend to more complex real-world problems where an optimal solution is not known. The problem uses a 3-input ( $x_1, x_2, x_3$ ), 1-output ( $y$ ) space. Known fuzzy rules, designed in advance, are used to generate test points of this function over the three-dimensional input space. The fuzzy classifier system is then required to “rediscover” this set of fuzzy rules using the function test points. For each variable, the domain of discourse is covered by three fuzzy sets (S, M, L), which are represented by equally spaced triangular membership functions in the range [0.0, 1.0]. The fuzzy classifiers used to generate the test data set are shown in Table 1.

Table 1. Fuzzy Rules used to Generate the Test Function Mapping

---

if ( $x_1=L$ ) $y = M$	//Most general rule
if ( $x_1=S$ ) & ( $x_3=S$ ) $y = L$	//Other general rules..
if ( $x_1=S$ ) & ( $x_3=L$ ) $y = S$	
if ( $x_2=M$ ) & ( $x_3=M$ ) $y = M$	
if ( $x_1=S$ ) & ( $x_2=S$ ) & ( $x_3=M$ ) $y = S$	//Specific rules....
if ( $x_1=S$ ) & ( $x_2=L$ ) & ( $x_3=M$ ) $y = L$	
if ( $x_1=M$ ) & ( $x_2=S$ ) & ( $x_3=S$ ) $y = M$	
if ( $x_1=M$ ) & ( $x_2=M$ ) & ( $x_3=S$ ) $y = L$	
if ( $x_1=M$ ) & ( $x_2=L$ ) & ( $x_3=S$ ) $y = L$	
if ( $x_1=M$ ) & ( $x_2=S$ ) & ( $x_3=M$ ) $y = S$	
if ( $x_1=M$ ) & ( $x_2=L$ ) & ( $x_3=M$ ) $y = L$	
if ( $x_1=M$ ) & ( $x_2=S$ ) & ( $x_3=L$ ) $y = S$	
if ( $x_1=M$ ) & ( $x_2=M$ ) & ( $x_3=L$ ) $y = S$	
if ( $x_1=M$ ) & ( $x_2=L$ ) & ( $x_3=L$ ) $y = M$	

---

Note that this classifier set contains one general rule with two “don’t cares”, three general rules with one “don’t care” and 10 specific rules. An approximately equivalent fuzzy decision table using 27 specific rules (i.e. not including “don’t cares”) is shown in Figure 1 and an attempt to show the shape of the test function in two-dimensions is shown in Figure 2.

x2	S M L		
x3	S M L		
S	L	L	L
M	S	M	L
L	S	S	S

x1 = S

x2	S M L		
x3	S M L		
S	M	L	L
M	S	M	L
L	S	S	M

x1 = M

x2	S M L		
x3	S M L		
S	M	M	M
M	M	M	M
L	M	M	M

x1 = L

Figure 1. Approximate Equivalent Representation of the Test Fuzzy Rule Base using only Specific Rules

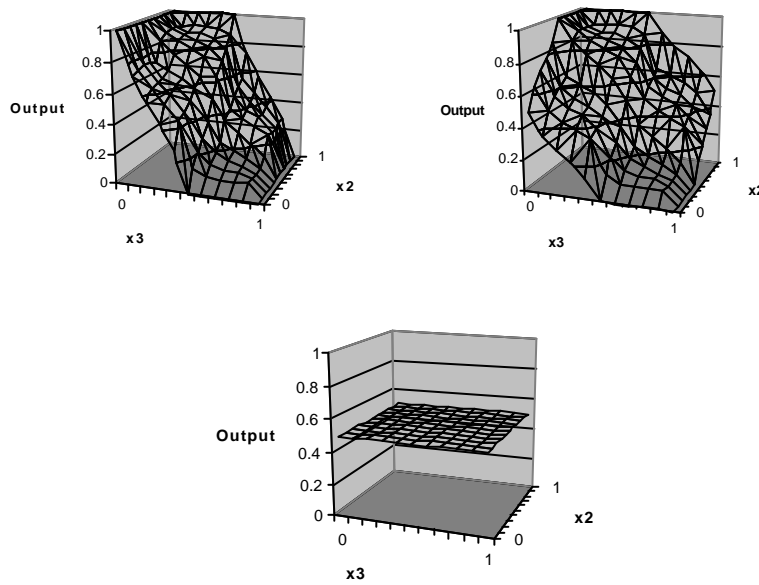


Figure 2. Functional Mapping for x1 = 0.0 (top-left), x1 = 0.5 (top-right) and x1 = 1.0 (bottom)

### 3.2 Fuzzy Rule Representation, Credit Assignment and Evolutionary Algorithm Details

The classifier syntax employed here is similar to that used in other studies (e.g. [9], [24]) where each classifier is represented by a string of integers. Each integer encodes a fuzzy set for a particular input variable. For example, if the input domain is covered by three fuzzy sets (SMALL, MEDIUM, LARGE) then '1' represents SMALL, '2' represents MEDIUM and '3' represents LARGE. A value of '0' in the antecedent of a classifier indicates a "don't care" condition. In an example three-input, single-output fuzzy classifier system, the string 120:1 represents the rule

IF (x1=SMALL) AND (x2=MEDIUM) THEN y = SMALL

where (x3=DON'T CARE) has been omitted for brevity. The classifier population is initially generated to contain random rules; and the number of rules is allowed to increase or decrease under the action of the GA.

At each learning step, the inputs are matched against the classifier list to form a Match Set, which contains all classifiers matched to degree greater than zero. This match set is divided into a number of subsets, where each subset contains classifiers with the same antecedent (taking into account "don't cares"). A single classifier is chosen from each subset stochastically, favouring classifiers with larger accrued strength to form the Fire Set. Fuzzy inference, aggregation and defuzzification are carried out using the Fire Set to generate the classifier system output. The environmental reward is calculated as the inverse of the absolute error, comparing the classifier system output with the known correct output. This reward is distributed to the classifiers in the Fire List proportional to classifier activations. If an input vector is encountered for which no matching fuzzy classifier exists, a cover operator is applied to generate a matching classifier with a random consequent. A trial consists of a complete run through the test data which covers the whole input space.

At the end of each trial a fitness value is calculated for each classifier. A niched GA is then applied in which classifiers with the same antecedent (including "don't cares") are formed into subpopulations. Roulette-wheel selection is then applied to each population to select parents for reproduction. Crossover is not employed. The following mutation operators are then applied to each parent to generate offspring:

CreepMutateLHS :	Increments/decrements a single antecedent fuzzy set label
MutateLHS:	Replaces an antecedent fuzzy set with a random value
CreepMutateRHS:	Increments/decrements a single consequent fuzzy set label
MutateRHS:	Replaces a consequent fuzzy set with a random value

These offspring are then inserted back into each subpopulation, replacing the weakest classifiers in the subpopulation if the subpopulation exceeds a certain size (a parameter varied in the experiments). The resulting subpopulations are then merged to form the complete fuzzy classifier set for the next generation.

### 3.3 Experimental Results

In these experiments, performance of fuzzy classifier systems which allow and do not allow “don’t cares” in the classifier syntax are compared. This allows us to study the effect of including general rules on system performance. In the case of classifier systems which allow “don’t cares” three different fitness evaluation methods are compared: strength-based fitness, accuracy-based fitness, and a combination of both. All graphs presented are the average of 20 GA runs starting with a different initial random seed.

#### Learning using only Specific Rules

Figure 3 shows the mean square error of the classifier system which does not employ “don’t cares”, versus generation number. Fitness evaluation is strength-based. The population is initialised to contain 40 random classifiers, and the population size is allowed to grow to a maximum of 80 under the action of the GA.

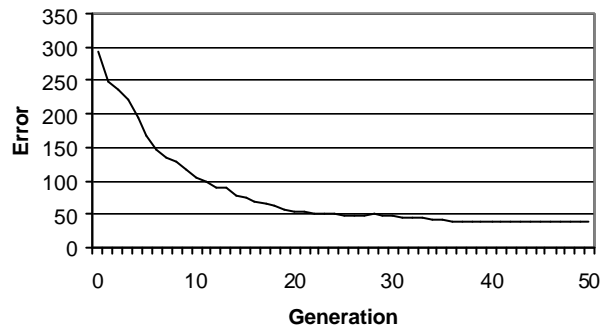


Figure 3. Error versus Generation using only Specific Rules

It was found that the classifier system consistently found the set of 27 specific rules described above in every case after around 40 generations and, in some cases, after only 20 generations. Note that the error never actually reduces to zero. This is because the collection of 27 specific rules is not exactly the same as the collection of 14 general and specific rules used to generate the test set. However, the solution found using specific rules is the best that can be achieved with this representation. To further investigate the consistency of the GA to find this optimal rule-set, the standard deviation of error over the ten runs was calculated. This is shown in Figure 4.

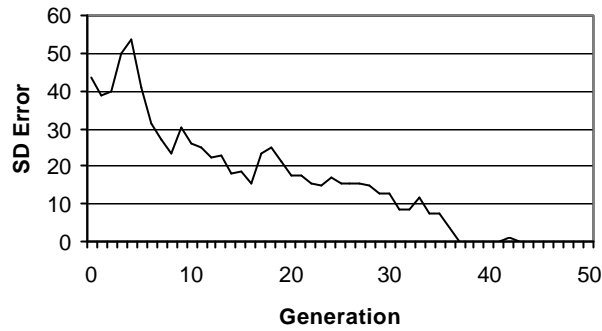


Figure 4. Standard Deviation of Error versus Generation using only Specific Rules

### Learning using General Rules – Strength Based Fitness

The same experiment, with the same population size, was carried out, but this time allowing “don’t cares” in the classifier syntax. Again, strength-based fitness is employed. The results are shown in Figure 5. Clearly the introduction of generalisation has degraded the performance of the classifier system, and when the best evolved rule-sets were inspected they were found to contain several rules which were clearly overgeneral, and therefore sub-optimal. In addition, by allowing “don’t cares”, we have effectively increased the search space which the GA has to cover.

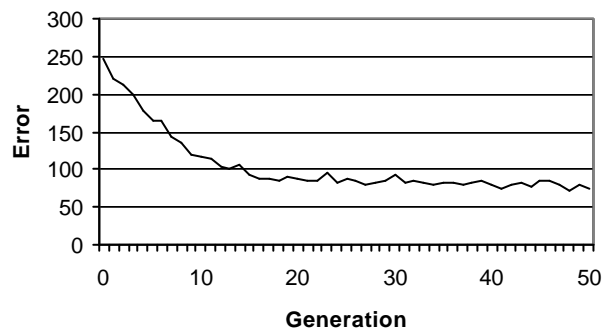


Figure 5. Mean Error versus Generation using Specific and General Rules

The GA runs using both specific and general classifiers were also much less consistent from run to run than using specific rules only. Figure 6 shows the standard deviation of error versus generation number in this case.

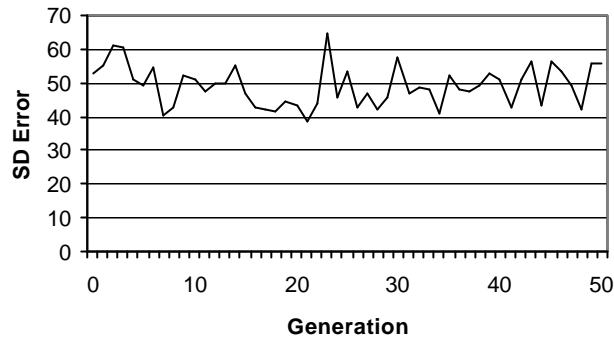


Figure 6. Standard Deviation of Error using Specific and General Rules

### Learning using General Rules – Accuracy Based Fitness

An attempt to emulate accuracy based-fitness was used by which the standard deviation of classifier payoff was calculated over each trial. The fitness of each classifier is calculated as the inverse of the standard deviation in payoff. The intention here is that classifiers which are inappropriately over-general will be penalised, since they will receive large variations in payoff. To take into account the additional size of the search space using “don’t cares” the initial population size was increased to 100 and allowed to grow up to a maximum of 200. The mean error versus generation number in this experiment is shown in Figure 7.

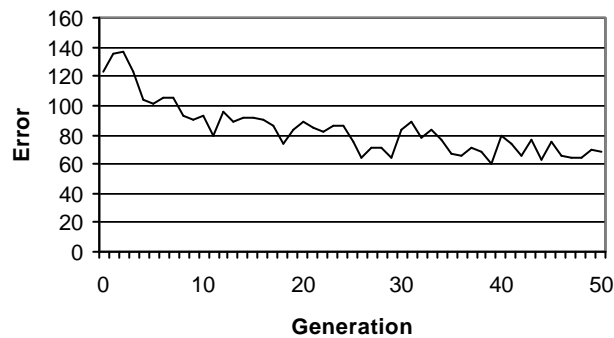


Figure 7. Error versus Generation Number using Accuracy-Based Fitness

Although error in the first few generations is lower than using strength-based fitness (most likely due to the larger population size), the overall performance is not significantly better.

### Learning using General Rules – Combined Strength/Accuracy Based Fitness

In this experiment each classifier fitness was calculated as its strength divided by the standard deviation in payoffs received over a trial. Results are shown in Figure 8.

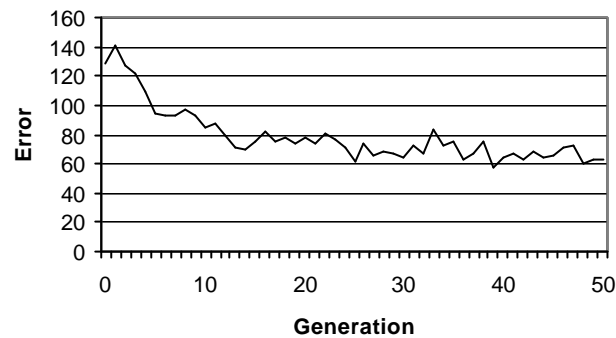


Figure 8. Error versus Generation Number using Strength/Accuracy Based Fitness

Once more, no significant increase in performance is observed.

### 3.4 Discussion

These experiments indicate that evolving the optimal collection of general and specific classifiers can be a difficult task, while evolving a sub-optimal set of specific classifiers appears much easier (at least on this simple problem). The accuracy-based experiments are particularly surprising since the evolution of optimally general classifier sets in the discrete valued classifier system appears much easier. There are several apparent reasons for this. Firstly, in a fuzzy classifier system several rules fire in parallel (this is how the system achieves interpolation); credit assignment is much more difficult in the fuzzy case and it may well be that apportioning credit in proportion to a fuzzy classifier's activation level is not appropriate. A further difficulty is measuring the accuracy of a rule's predicted payoff since (particularly early in the search) a fuzzy rule will fire with many different other fuzzy rules at different time-steps, giving very different payoffs. Yet another difficulty is that the payoff a fuzzy rule receives depends on the input vector – an active fuzzy rule will receive different payoffs for different inputs. This further complicates payoff predictions used as the basis for accuracy based fitness. Two possible ways forward to overcome these problems might be (1) to have a rule learn payoff as a form of  $R^N \rightarrow R$  mapping, e.g. using a neural network where the network inputs are the input variable values and the output is the payoff, and (2) use some form of rule tagging to discover high-performance "corporations" of fuzzy rules which, when they fire together, provide high payoff.

## 4. Conclusions

This contribution has highlighted the difficulties in incorporating generalisation into the fuzzy classifier system. A simple test problem has been introduced specifically for testing the ability of a fuzzy classifier system to learn an optimal combination of general and specific rules. Initial experiments using both strength- and accuracy-based fitness have demonstrated how difficult even this simple problem is for a basic fuzzy classifier system. Clearly much further work is required on the issue of generalisation in fuzzy classifier systems, to bring our level of understanding on a par with that of discrete-valued classifier systems.

## References

1. Carse B., Fogarty T.C. and Munro A. (1996). Evolving Fuzzy Rule-based Controllers using Genetic Algorithms. *Fuzzy Sets and Systems* 80(3), pp.273-293.
2. Carse B., Pipe A.G., Davies O. (1997). Parallel evolutionary learning of fuzzy rule bases using the island injection genetic algorithm, *Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernetics*, pp. 3692-3697.
3. Bonarini A. (2000). An Introduction to Learning Fuzzy Classifier Systems. In P.L. Lanzi, W. Stolzmann and S.W. Wilson (Eds.), *Learning Classifier Systems- from Foundations to Applications*, *Lecture Notes in Artificial Intelligence*, pp. 83-104. Springer Verlag Berlin Heidelberg, Germany.
4. Cordón O., Herrera F., Hoffmann F. and Magdalena L. (2001). *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. *Advances in Fuzzy Systems – Applications and Theory* Vol. 19, World Scientific.
5. Valenzuela-Rendón M. (1991). The Fuzzy classifier system: a classifier system for continuously varying variables. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 346-353, San Mateo, CA:Morgan Kaufman.
6. Valenzuela-Rendón M. (1998). Reinforcement learning in the fuzzy classifier system. *Expert Systems with Applications* 14, pp. 237-247.
7. Parodi A. and Bonelli P. (1993). A New approach to fuzzy classifier systems. In *Proceedings of 5th International Conference on Genetic Algorithms* pp. 223-230. San Mateo, CA:Morgan Kaufman.
8. Ishibuchi H., Nakashima T. and Murata T. (1999) Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics* 29, pp. 601-608.
9. Bonarini A. (1996), Evolutionary learning of fuzzy rules: competition and cooperation. In W. Pedrycz (Ed.), *Fuzzy Modelling: Paradigms and Practice*, pp. 265-284, Norwell, MA: Kluwer Academic Press.
10. Wilson S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computing* 3(2), pp.149-175.

11. Cordon O., Herrera F. and Zwir I. (2002) Linguistic modelling by hierarchical systems of linguistic rules. *IEEE Transactions on Fuzzy Systems*, 10 (1) pp. 2-20.
12. Kim J. and Zeigler B.P. (1996) Designing fuzzy logic controllers using a multi-resolutional search paradigm. *IEEE Transactions on Fuzzy Systems* 4, pp.213-216
13. Cheong F. and Lai R. (2000) Constraining the optimisation of a fuzzy logic controller using an enhanced genetic algorithm. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics* 30, pp. 31-46.
14. González A., Pérez R. and Verdegay J.L. (1993) Learning the structure of a fuzzy rule: a genetic approach. In *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies (EUFIT'93)*, Aachen, Germany, pp. 57-70.
15. González A. and Pérez R. (1999) SLAVE: A genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems* 7(2), pp. 176-191.
16. Alba E., Cotta C. and Troya J.M. (1999). Evolutionary design of fuzzy logic controllers using strongly typed GP. *Mathware and Soft Computing* 6(1), pp. 109-124.
17. González A. and Pérez R. (2001). Selection of relevant features in a fuzzy genetic learning algorithm. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics* 31(3), pp. 417-425.
18. Hoffmann F. and Pfister G. (1996). Learning of a fuzzy control rule base using messy genetic algorithms. In: F. Herrera and J.L. Verdegay (Eds.), *Genetic Algorithms and Soft Computing*, No 8 in *Studies in Fuzziness and Soft Computing*, pp. 279-305, Physica-Verlag.
19. Hoffmann F. and Pfister G. (1997). Evolutionary design of a fuzzy knowledge base for a mobile robot. *International Journal of Approximate Reasoning* 17(4), pp. 447-469.
20. Holland J.H. 1988. Escaping Brittleness: The Possibilities of General Purpose Machine Learning Algorithms applied to Parallel Rule-based systems. In: Michalski R.S., Carbonell J.G and Mitchell T.M. (Eds.), *Machine Learning: an Artificial Intelligence Approach*, vol.2. Kaufmann, Los Altos, Calif, 1988.
21. Kovacs T. (1996). Evolving optimal populations with XCS classifier systems. Master's Thesis, School of Computer Science, University of Birmingham, Birmingham, UK.
22. Lanzi P.L. (1999). An Analysis of generalisation in the XCS classifier system. *Evolutionary Computation* 7(2), pp. 125-149.
23. Goldberg D.E., Korb B. and Deb K. (1989). Messy genetic algorithms: motivation, analysis and first results. *Complex Systems* 3, pp.493-530.
24. Pipe A.G. and Carse B (2000) Autonomous acquisition of fuzzy rules for mobile robot control: first results from two evolutionary computation based approaches. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'00)*, pp 849-856, Morgan-Kaufman, San Francisco, CA.