# Evolving Fuzzy Detectives:
# An Investigation into the Evolution of Fuzzy Rules

**Peter J. Bentley**

Department of Computer Science, University College London,
Gower Street, London WC1E 6BT, UK
P.Bentley@cs.ucl.ac.uk

**Abstract:** This paper explores the use of genetic programming to evolve fuzzy rules for the purpose of fraud detection. The fuzzy rule evolver designed during this research is described in detail. Four key system evaluation criteria are identified: intelligibility, speed, handling noisy data, and accuracy. Three sets of experiments are then performed in order to assess the performance of different components of the system, in terms of these criteria. The paper concludes: 1. that many factors affect accuracy of classification, 2. intelligibility and processing speed mainly seem to be affected by the fuzzy membership functions and 3. noise can cause loss of accuracy proportionate to the square of noise.

## 1 Introduction

It is easy to spot a thief if he wears a black mask and carries a bag over his shoulder with the word 'swag' written boldly across it. Most crimes are perhaps not as obvious as this amusing movie cliché, but there will be, more often than not, physical evidence left behind which will incriminate the wrongdoer. However, there is a type of crime far subtler in its implementation. This clandestine activity relies upon deception, concealment and mendacity. It is known as *fraud*, and it impacts on every aspect of our financial world - from insurance to social security benefits to pensions. Its exposure requires a different type of detection. There are no fingerprints left to be found by forensics - but in the computer databases there are other types of fingerprints stored unknowingly by the fraudsters. These fingerprints are small patterns of data, hidden amongst vast archives of information.

Until recently, often the only way to identify such fraud was for experts to study each data item and mentally apply a set of learned rules. For example, if the home insurance claim is for a large amount and if there have been many such claims from that address in a short space of time, then perhaps the claim is fraudulent. Identifying such 'fingerprints' in data is a laborious and slow task, and is dependent on the fingerprint

characterising a true fraudulent activity. However, with the advent of data mining and machine learning techniques, such detection can now be performed by computer.

This paper explores the use of evolutionary computation and fuzzy logic in combination to perform machine learning or *pattern classification*, with an emphasis on the capabilities required for the detection of fraud. The next section briefly describes the background to this area. Section 3 gives details of the fuzzy rule evolver and its different components. Section 4 outlines the evaluation requirements identified after consultation with our collaborating company. The fifth section describes and analyses three sets of experiments that were performed on the system and section 6 concludes.

# 2 Background

Machine Learning, pattern classification and data mining are huge fields in Computer Science, with countless different techniques in use or under investigation. This paper concentrates on a single approach: the use of *fuzzy logic* with *genetic programming* to classify data.

Fuzzy sets were introduced by Lofti Zadeh in 1965 [15]. Designed to allow the representation of 'vagueness' and uncertainty that conventional set theory disallowed, the sets and their manipulation by logical operators led to the development of the field known as Fuzzy Logic [3]. Despite the name, fuzzy techniques are actually capable of greater precision compared to classical approaches [6]. Fuzzy controllers have been used with considerable success: examples include controllers for elevators, subway trains, and even fuzzy autofocus systems for cameras [11].

Another appeal of fuzzy logic is its *intelligibility*. Fuzzy rules use linguistic identifiers such as 'high', 'short' and 'inexpensive'. Because all humans tend to think in such vague terms, the specification and understandability of such rules becomes simple, even to someone unaware of the mechanisms behind this technique [6]. The combination of representation of uncertainty, precision, and intelligibility has motivated the use of fuzzy logic in pattern classification [3], and indeed, forms the motivation for its use in this research.

Fuzzy logic can be combined or hybridized with many other techniques, including evolutionary algorithms. Some have developed fuzzy-evolutionary systems [12] where fuzzy logic is used to tune parameters of an evolutionary algorithm. Others use evolutionary-fuzzy approaches, where evolution is employed to generate or affect fuzzy rules [9,10]. This paper describes the latter approach, and makes use of Genetic Programming (GP).

John Koza developed GP for the purposes of automatic programming [7] (making computers program themselves). GP differs from other EAs in three main respects: solutions are represented by tree-structures, crossover normally generates offspring by concatenating random subtrees from the parents, and solutions are evaluated by *executing* them and assessing their function.

Like all evolutionary algorithms (EAs), GP maintains *populations* of solutions. These are evaluated, the best are selected and 'offspring' that inherit features from their 'parents' are created using crossover and mutation operators. The new solutions are then

evaluated, the best are selected, and so on, until a good solution has evolved, or a specific number of generations have passed.

EAs are often used for pattern classification problems [8], but although the accuracy can be impressive, it is often difficult to understand the evolved method of classification. By evolving fuzzy rules it is possible to get the best of both worlds - accurate and intelligible classification [9].

# 3 The Fuzzy Rule Evolver

The system developed during this research comprises two main elements: a Genetic Programming (GP) search algorithm and a fuzzy expert system. Figure 1 provides an overview of the system.

Data is provided to the system in the form of two comma-separated-variable (CSV) files: training data and test data. In each file the first row contains the labels for each column (e.g. "Length, Width, Height, Cost"). These labels are used by the system in the rules that it evolves. The next *Sn* rows (or *data items*) are known to be of the 'suspicious' class, the remaining data comprises one or more other classes. There may be up to 256 values in each data item, as long as the training and test data sets are consistent (normally training and test data files are constructed by splitting a single data file into two). The system assumes that the data is numerical and that there are no missing values (a data pre-processing program has been developed to assign numbers for alphanumeric entries and to fill missing values with random values). All training takes place on the training data set; testing of evolved rules takes place on both training and test data sets.

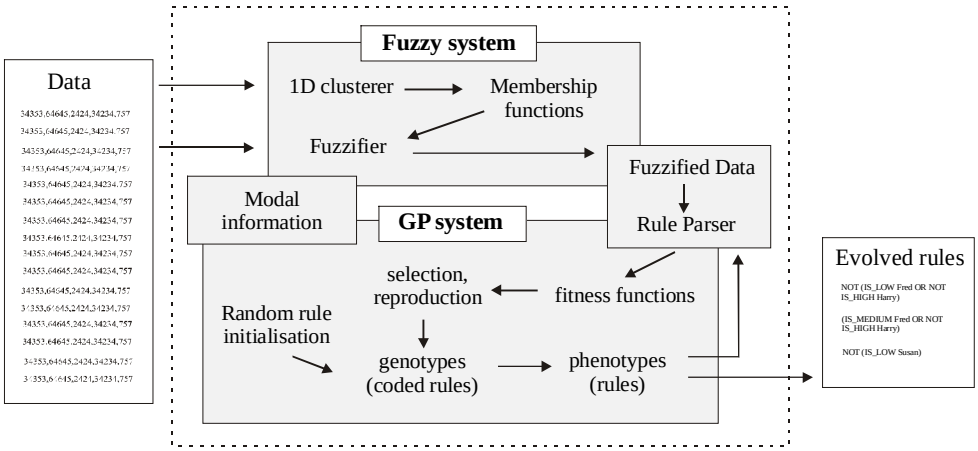


**Figure 1** Block diagram of the Evolutionary-fuzzy system.

## 3.1 Clustering

When started, the system first clusters each column of the training data into three groups using a one-dimensional clustering algorithm. A number of clusterers are implemented in the system, including C-Link, S-Link, K-means [5] and a simple numerical method (in which the data is sorted, then simply divided into three groups with the same number of items in each group). This paper investigates the last two of these methods in the system. Once selected by the user, the same clusterer is used for all learning and testing of the data.

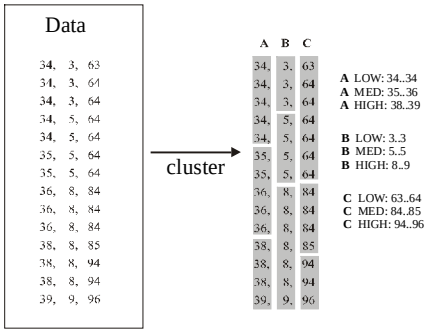


**Figure 2:** Data is clustered column by column to find the fuzzy membership function ranges.

After every column of the data has been successfully clustered into three, the minimum and maximum values in each cluster are found, see fig. 2. These values are then used to define the domains of the membership functions of the fuzzy expert system.

## 3.2 Fuzzy Membership Functions

Three membership functions, corresponding to the three groups generated by the clusterer, are used for each column of data. Each membership function defines the 'degree of membership' of every data value in each of the three fuzzy sets: 'LOW', 'MEDIUM' and 'HIGH' for its corresponding column of data. Since every column is clustered separately, with the clustering determining the domains of the three membership functions, every column of data has its own, unique set of three functions.
The system can use one of three types of membership function: 'non-overlapping', 'overlapping', and 'smooth', see figure 3. The first two are standard trapezoidal functions, the third is a set of functions based on the arctangent of the input in order to

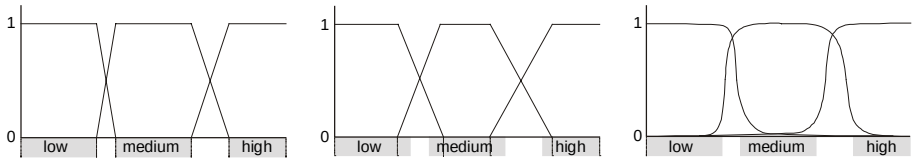


**Figure 3:** The three types of membership functions used by the system: non-overlapping (left), overlapping (middle), smooth (right).

provide a smoother, more gradual set of 'degree of memberships'.

Each type of membership function uses the results from the clusterer in a different way to determine the domains of the functions. The 'non-overlapping' functions give a membership of 1.0 for the fuzzy set corresponding to the cluster that the value falls in, and 0.0 for all other fuzzy sets. For example, a value that falls in the lowest cluster would be fuzzified into (1.0, 0.0, 0.0) for the low, medium and high fuzzy sets, respectively. Since, for this application, it is normal for all values to fall within one of the three clusters, it is extremely rare for values to be between two clusters. Hence, the 'non-overlapping' functions almost never allow a value to be a member of more than one fuzzy set at a time (i.e. the fuzzy sets are, to all intents and purposes, non-overlapping). In contrast, the 'overlapping' functions place the 'knees' and 'feet' of each function at three quarters of the values provided by the clusterer, see figure 3 (middle). This has the effect of ensuring that the three fuzzy sets overlap - a value towards the outer extent of the low fuzzy set might thus be fuzzified into (0.8, 0.2, 0.0) for low, medium and high fuzzy sets respectively. Finally, the 'smooth' functions increase the level of overlap still further. For example, a value in the centre of the low fuzzy set might be fuzzified into (0.98 0.02, 0.0), while a value towards the outer extent of the low fuzzy set might be fuzzified as (0.96, 0.4, 0.0).

Whichever set of membership functions are selected, they are then shaped according to the clusterer and used to fuzzify all input values, resulting in a new database of fuzzy values. The GP engine is then seeded with random genotypes (coded rules) and evolution is initiated.

## 3.3 Evolving Rules

The implementation of the GP algorithm is perhaps best described as a genetic algorithmist's interpretation of GP, since it employs many of the techniques used in GAs to overcome some of the problems associated with simple GP systems. For example, this evolutionary algorithm uses a crossover operator designed to minimise the disruption caused by standard GP crossover, it uses a multiobjective fitness ranking method to allow solutions which satisfy multiple criteria to be evolved, and it also uses binary genotypes which are mapped to phenotypes.

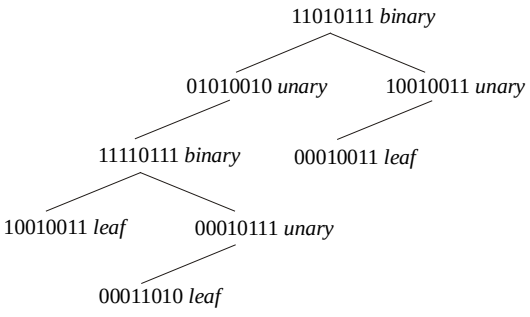


**Figure 4:** An example genotype used by the system.

***Genotypes and Phenotypes***

Genotypes consist of variable sized trees, where each node consists of a binary number and a flag defining whether the node is binary, unary or a leaf, see figure 4. At the start of evolution, random genotypes are created (usually containing no more than 3 binary and 4 unary nodes). Genotypes are mapped onto phenotypes to obtain fuzzy rules, e.g. the genotype shown in fig. 4 maps onto the phenotype:

"(IS_MEDIUM (Height OR IS_LOW Age) AND IS_MEDIUM Age)".

Currently the system uses two binary functions: 'OR' and 'AND', four unary functions: 'NOT', 'IS_LOW', 'IS_MEDIUM', 'IS_HIGH', and up to 256 leaves (column labels such as "Length", "Width", "Height", "Cost"). Depending on the type of each node, the corresponding binary value is mapped to one of these identifiers and added to the phenotype.

***Rule Evaluation***

Every evolved phenotype (or fuzzy rule) is evaluated by using the fuzzy expert system to apply it to the fuzzified training data, resulting in a defuzzified score between 0 and 1 for every fuzzified data item. (Section 3.4 describes this in full.) This list of scores is then assessed by four fitness functions which provide separate fitness values for the phenotype, designed to:

i. minimise the number of misclassified items (where a misclassified item is a 'normal' data item with a score > 0.5).
ii. maximise the difference between the average scores for correctly classified 'suspicious' items and the average scores for 'normal' items (where a correctly classified suspicious item is a data item in the first *Sn* of the training set with score > 0.5).
iii. maximise the sum of scores for 'suspicious' data items.
iv. penalise the length of any rules that contain more than four identifiers (binary, unary, or leaf nodes).

The first function ensures as few misclassifications as possible. The second forces evolution to distinguish between 'suspicious' and 'normal' classes of data, while the third demands that 'suspicious' items are given higher scores than 'normal' ones. The final function ensures that all evolved rules are short - serving the dual purpose of preventing bloat and increasing the readability of the final output.

***Rule Generation***

Using these four fitness values for each rule, the GP system then employs the SWGR multiobjective optimisation ranking method [2] to determine how many offspring each pair of rules should have. (Fitnesses are scaled using the effective ranges of each function, multiplied by importance values and aggregated. Rules with higher overall fitnesses are given higher ranking values, and hence have an increased probability of producing offspring.) Child rules are generated using one of two forms of crossover. The first type of crossover emulates the single-point crossover of genetic algorithms by finding two random points in the parent genotypes that resemble each other, and splicing the genotypes at that point. By ensuring that the same type of nodes, in approximately the

same places, are crossed over, and that the binary numbers within the nodes are also crossed, an effective exploration of the search space is provided without excessive disruption [1]. The second type of crossover generates child rules by combining two parent rules together using a binary operator (an 'AND' or 'OR'). This more unusual method of generating offspring (applied approximately one time out of every ten instead of the other crossover operator) permits two parents that detect different types of 'suspicious' data to be combined into a single, fitter individual. Mutation is also occasionally applied, to modify randomly the binary numbers in each node by a single bit.

The GP system employs population overlapping, where the worst *Pn*% of the population are replaced by the new offspring generated from the best *Pm*%. Typically values of *Pn* = 80 and *Pm* = 40 seem to provide good results. The population size was normally 100 individuals.

***Modal Evolution***

Each evolutionary run of the GP system (usually only 15 generations) results in a short, readable rule which detects some, but not all, of the 'suspicious' data items in the training data set. Such a rule can be considered to define one mode of a multimodal problem. All items that are correctly classified by this rule (recorded in the modal database, see figure 1) are removed and the system automatically restarts, evolving a new rule to classify the remaining items. This process of modal evolution continues until every 'suspicious' data item has been described by a rule. However, any rules that misclassify more items than they correctly classify are removed from the final rule set by the system.

## 3.4 Assessment of Final Rule Set

Once modal evolution has finished generating a rule set, the complete set of rules (joined into one by disjunction, i.e., 'OR'ed together) is automatically applied to the training data and test data, in turn. Information about the system settings, number of claims correctly and incorrectly classified for each data set, total processing time in seconds, and the rule set are stored to disk.

## 3.5 Applying Rules to Fuzzy Data

The path of evolution through the multimodal and multicriteria search space is guided by fitness functions. These functions use the results obtained by the Rule Parser - a fuzzy expert system that takes one or more rules and interprets their meaning when they are applied to each of the previously fuzzified data items in turn.

This system is capable of two different types of fuzzy logic rule interpretation: traditional fuzzy logic, and *membership-preserving* fuzzy logic, an approach designed during this research. Depending on which method of interpretation has been selected by the user, the meaning of the operators within rules and the method of defuzzification is different.

***Traditional Fuzzy Logic Rule Parser***

Traditional fuzzy logic involves finding 'degrees of membership' in the fuzzy sets for each value in the current data item, then using operators to select which membership value should be selected and used in combination. So, given a data item comprising two fuzzified values:

A(0.0, 0.2, 0.8)

B(0.1, 0.9, 0.0)

and a fuzzy rule:

(IS_LOW A AND IS_MEDIUM B)

the traditional fuzzy rule parser takes the degree of membership of A for fuzzy set LOW and the degree of membership of B for the fuzzy set MEDIUM, and calculates which of the two is smaller. So in this case, the result of applying the rule is 0.0. Table 1 describes the behaviour and syntax of each of the fuzzy operators.

**Table 1:** Traditional fuzzy operators.

| **Operator** | **Result** |
|---|---|
| IS_LOW <a, b, c> | A |
| IS_MEDIUM <a, b, c> | B |
| IS_HIGH <a, b, c> | C |
| NOT a | 1-a |
| (a AND b) | min(a,b) |
| (a OR b) | max(a,b) |

This fuzzy grammar imposes certain constraints upon allowable solutions. For example, the argument to 'IS_LOW', 'IS_MEDIUM' or 'IS_HIGH' must always consist of a fuzzy vector: $<Low_{membership}, Medium_{membership}, High_{membership}>$. The arguments to 'AND', 'OR' and 'NOT' functions must always be single-valued results obtained from the application of one or more of the functions.

As is clear from the example phenotype given in section 3.3.1, evolved rules do not always satisfy the constraints imposed by fuzzy grammars. However, rather than impose these damaging constraints on evolution, such grammatically incorrect rules are corrected by the rule parser. (Work performed during this research showed that using mapping to satisfy constraints in a GP system is one of the more effective approaches [14].)

**Table 2:** Mapping performed by the Rule Parser.

| **Operator** | **Result** |
|---|---|
| <a, b, c> | IS_HIGH <a, b, c> |
| IS_LOW a | a |
| IS_MEDIUM a | a |
| IS_HIGH a | A |

Functions requiring a fuzzy vector, but receiving a single value do nothing. Functions requiring a single value, but receiving a fuzzy vector, apply 'IS_HIGH' by default in order to generate the single value. Table 2 describes this behaviour in full. Consequently, when interpreted by the fuzzy rule parser, the rule in section 3.3.1 equates to:

"((IS_HIGH Height OR IS_LOW Age) AND IS_MEDIUM Age)".

Defuzzification of the final output value is unnecessary (although it is possible to impose a scaling, or non-linear function to transform the output in some way). It was decided simply to use a one-to-one function for defuzzification (i.e., return the output of the fuzzy rule as the defuzzified value).

***Membership-Preserving Fuzzy Logic Rule Parser***

The alternative behaviour of the rule parser preserves the three membership values within data items, even after the application of operators such as 'IS_LOW'. This is done in an attempt to permit rules to use all the information found by the clusterer, and thus hopefully to reduce the number of rules needed to classify data. In addition, the operators are designed to be more conducive to evolution by allowing multiple operators to have combined effects without constraints on syntax. For example:

IS_HIGH IS_HIGH **v**

is now equivalent to

IS_VERY_HIGH **v**

It should be noted, however, that the English descriptors for these operators does not always fully encompass their behaviour in a rule. Table 3 shows the new behaviours of the operators.

**Table 3:** Membership-preserving fuzzy operators.

| **Operator** | **Result** |
|---|---|
| <a, b, c> | <a, b, c> |
| IS_LOW <a, b, c> | Conc. <c, b, a> |
| IS_MEDIUM <a, b, c> | Conc. <0, max(a,c), b> |
| IS_HIGH <a, b, c> | Conc. <a, b, c> |
| NOT <a, b, c> | <c, b, a> |
| (<a, b, c> AND <d, e, f>) | min(<a, b, c>,<d, e, f>) |
| (<a, b, c> OR <d, e, f>) | max(<a, b, c>,<d, e, f>) |

Where Conc. *concentrates* the vector (making the largest value larger and the other two values smaller).

Because this novel approach preserves all three membership values during the application of all operators (although the values may be intensified or reduced), the final result is also a vector comprising three values. To obtain a single, defuzzified value, three defuzzification functions are applied, using the vector to define three trapezoidal shapes, see figure 5. The shapes are then 'piled up on top of each other' and the centre of mass calculated (using overlapping shapes results in a loss of information). A centre of mass falling in the centre results in an output of 0.5, falling to the right gives a score between

0.5 and 1.0, and if the centre of mass falls to the left, the final defuzzified value is between 0.5 and 0, see figure 6.

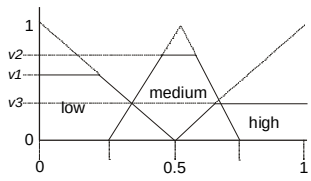


**Figure 5:** Defuzzifying the three membership values <*v1, v2, v3*>

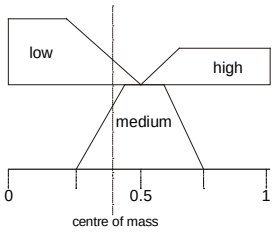


**Figure 6:** Finding the centre of mass during defuzzification.

The membership-preserving (M-P) fuzzy logic is designed to make use of overlapping membership functions. Indeed, for non-overlapping functions, the behaviour of the M-P fuzzy operators becomes largely identical to the traditional operators.

# 4 Criteria for Fraud Detection

The research described here is being carried out with the eventual aim of the detection of 'suspicious' home insurance claims. This difficult real-world classification task does not simply involve finding the most accurate method for distinguishing between ordinary and dubious data items. There are, in fact, more important criteria for evaluating the performance of a technique. Table 4 shows the four capabilities considered to be most important by our collaborating company Lloyds/TSB, with importance rankings.

**Table 4:** Important features of a good fraud-detection system.

| **Feature** | **Importance** |
|---|---|
| Intelligibility of classification rules | 1 |
| Speed of classification | 2 |
| Handling noisy data | 2 |
| Accuracy of classification | 3 |

It may be surprising to note that accuracy is considered less important than intelligibility. However, for this type of application, an expert must review all suggestions made by the classifier (wrongly accusing anyone of fraud is a serious and potentially libellous activity, so the computer should be used only to identify the possibility of suspicion to experts). If the person cannot find an easily understandable explanation of why a particular data item has been labelled as 'suspicious', then the result is of little use, regardless of the reported accuracy of classification.

Speed of classification is also essential, for most real-world financial problems of this type involve an enormous quantity of data. Increasingly it is becoming necessary for learning techniques to be performed in real time (as new data arrives), but at the very

least, the detection method must be fast enough to keep up to date, and also fast enough to justify its use at all.

The ability to handle noisy data was ranked equal in importance with speed. Input errors, omitted data, or conversion problems may cause noise in the data. Although such noise is unlikely to affect more than a small percentage of values in the data, it is clearly important that the classifier is not misled by any occurrence of noise.

Other important considerations include minimising the misclassifications by the system - it is considered better to miss a few dubious data items than to misclassify normal data. It is clearly not good for customer relations if too many people are wrongly investigated for potential wrong-doing. This is the reason for the inclusion of the first fitness function, described in section 3.3.2.

# 5 Experiments

## 5.1 Objectives

With the requirements for a good fraud-detection system in mind, this section describes a series of experiments designed to evaluate these key capabilities of the system.

The experiments investigate three aspects of the system: the effect of using different membership functions and fuzzy operators, the effect of using different clusterers, and the ability of the system to cope with noisy data. For all three sets of experiments, the intelligibility of results, processing time, and accuracy of detection are assessed.

## 5.2 Experimental Setup

To allow comparison of this system with other techniques reported in the literature, the fuzzy rule evolver was applied to two standard data sets for all experiments: the Iris and Wisconsin Breast Cancer data sets.

The Iris data is "perhaps the best known database to be found in the pattern recognition literature"[1], and comprises a simple domain of 150 instances in three classes, each of 50 items. Data items have four attributes; there are no missing values. Because the 'Setosa' class is linearly separable from the other two classes, for all experiments the system was set the harder task of detecting the 'Virginica' class from the 'Versicolour' and 'Setosa' classes combined. Training and test data files were prepared by splitting the data set into two (taking alternate data items for each file). Misclassification rates for this data set are normally reported as 0% for the 'Setosa' class and "very low" for the other classes in the literature e.g. [4].

The Wisconsin Breast Cancer data is a more complex data set, comprising 699 instances in two classes: 'Malignant' (241 data items) and 'Benign' (458 items). There are 16 missing values in the data, which were filled by random numbers. The training and test data sets were constructed by splitting the file into two, taking alternate values. (For the sake of symmetry, one 'Malignant' item was discarded and two 'Benign' items

[1] According to the information provided by UCI with the data.

duplicated, resulting in two sets of 350 data items, each with 120 'Malignant'.) Results reported in the literature include accuracies of 93.5%, 95.9% [13], and 92.2% [16].

50 trials were run for each experiment, with the average and best accuracies reported here. Percentage accuracy of detection was found by calculating:

$$100 - \frac{100(MisclassifiedItems + Unclassifieditems)}{TotalItems}$$

Intelligibility was measured in terms of the average number of rules evolved - the fewer the rules, the more intelligible the result. Average processing speed was measured in seconds (and includes the negligible time taken to apply the completed rule set to both data sets).

The fitness functions reported in section 3.3.2 were used without change for all experiments. Importance rankings [2] were set as 0.5, 2.0, 1.0 and 0.5 for fitness functions one to four, respectively. Mutation of a single bit occurred with a probability of 0.001 in each genotype. Population sizes of 100 were used, and each modal evolutionary run was for exactly 15 generations. The K-Means clusterer was used in the system (unless otherwise stated). Experiments were run on a PC with a 233Mhz AMD K6 processor.

## 5.3 Experiment 1: Investigating the Effect of Membership Functions

The objective of the first set of experiments was to examine the effects of different membership functions (and different ways of using the information contained in the membership functions) on the ability of the system to detect data items with good intelligibility, speed, and accuracy. Four different system set-ups were investigated: traditional fuzzy logic with non-overlapping and overlapping membership functions, and M-P fuzzy logic with overlapping and smooth membership functions. (Traditional fuzzy logic does not work well with the level of overlap provided by the smooth functions, and the M-P fuzzy logic with non-overlapping functions behaves in the same way as traditional fuzzy logic, so these set-ups are not investigated here.) Table 5 shows the results obtained from 50 runs of each system set-up for each data set.

As shown in Table 5, for both data sets the average accuracy appears to fall as the level of overlap of membership functions is enlarged. However, there is clearly a quite dramatic increase in intelligibility (a reduced number of rules) as the overlap increases. This is illustrated by two example solutions evolved by the system for the Wisconsin Breast Cancer data set. Figure 7 shows a typical 12-rule set evolved when using traditional fuzzy logic and non-overlapping membership functions. Figure 8 shows a typical single rule evolved when using M-P fuzzy logic with smooth membership functions. It should be apparent that the latter is substantially more intelligible than the former. Not only that, but by reducing the number of rules, far more effective feature-selection takes place (i.e., instead of using all ten fields in the data, the single rule shows that only two are required).

**Table 5:** Mean and best accuracy rates, processing times and intelligibility of solutions when using different membership functions and fuzzy operators. (Accuracy values in normal intensity indicate results for the training set, bold values show results for the test data set.)

| | **Iris data** | | | | **Cancer data** | | | |
|---|---|---|---|---|---|---|---|---|
| **System:** | **Av. Accuracy** | **Best accuracy** | **Av. time** | **Av. # of rules** | **Av. accuracy** | **Best accuracy** | **Av. time** | **Av. # of rules** |
| FL, non-overlapping MFs | 96.67%<br>**95.79%** | **97.3%** | 25.6 secs | 2.94 | 98.6%<br>**94.07%** | **96.0%** | 317 secs | 9.88 |
| FL, overlapping MFs | 91.3%<br>**94.69%** | **96%** | 13.6 secs | 1.50 | 93.01%<br>**90.44%** | **96.29%** | 335 secs | 7.16 |
| M-P FL, overlapping MFs | 96.05%<br>**90.67%** | **90.67%** | 15.1 secs | 1.04 | 91.19%<br>**86.93%** | **92.57%** | 175 secs | 4.58 |
| M-P FL, overlapping smooth MFs | 82.69%<br>**82.59%** | **88%** | 16.2 secs | 1 | 95.14%<br>**95.71%** | **95.71%** | 162 secs | 1 |

It is clear that accuracy is reduced as the number of rules that classify the data is reduced. The exception to this is the MP-FL system with smooth MFs applied to the cancer data, which generated both accurate results with a very intelligible single (and simple) rule, e.g. fig 8. However, this result seems likely to be more the exception than the rule – the accurate result may well be due to a fortunate combination of placement of membership functions, and the combination of the three fuzzy membership values for this particular problem. Nevertheless, the result certainly indicates that it is possible to classify real-world problems with both accuracy and intelligibility.

```
Adhesion
(ClumpThickness AND CellShape)
(CellSize AND Chromatin)
(ClumpThickness AND EpithCellSize)
(CellSize AND ClumpThickness)
(IS_LOW Samplecode AND BareNuclei)
(IS_MEDIUM NormalNucleoli AND ClumpThickness)
(BareNuclei AND EpithCellSize)
(Mitoses AND ClumpThickness)
(ClumpThickness AND BareNuclei)
(IS_MEDIUM NormalNucleoli AND BareNuclei)
(ClumpThickness AND IS_LOW EpithCellSize)
```

**Figure 7:** A 12-rule set evolved using traditional fuzzy interpretation by the rule parser and non-overlapping functions.

```
IS_HIGH (CellSize OR BareNuclei)
```

**Figure 8:** A single rule evolved using M-P fuzzy interpretation by the rule parser and smooth functions.

As Table 5 shows, processing times fell as the level of overlap of membership functions was increased. This speedup is readily explainable: as the overlap of MFs was increased, the number of rules evolved by the system fell, and since each rule is the result of one modal evolutionary run of 15 generations, system speed is proportionate to the number of rules evolved during classification. The longest learning time for the 7000-value Cancer set took around five and half minutes in these experiments. However, once learned, the time taken to apply the rules to the data is less than one second.

## 5.4 Experiment 2: Investigating the Effects of Clusterers

The objective of this second set of experiments was to determine the impact of using different clusterers in terms of the three performance measures of intelligibility, speed and accuracy. Two extremes of clusterer were employed: the basic method (described in section 3.1) and the substantially more advanced K-Means approach. For these tests, the system used traditional fuzzy logic and non-overlapping membership functions.

Tables 6 shows the results obtained from 50 runs of each system set-up for each data set. As can be seen from the average and best accuracy percentages, the basic clustering does result in slightly reduced performance of classification. The performance loss is perhaps surprisingly low, though, when it is recalled how simple the basic clustering method is, compared to the K-Means approach. Different rules and different numbers of rules were evolved when using each type of clusterer, as shown by the other results in Table 6. There does not seem to be any clear correlation between intelligibility or processing speed and the type of clusterer used.

## 5.5 Experiment 3: Investigating the Effects of Noise on the System

The objective of this final set of experiments was to evaluate the change of performance of the system as levels of noise in the data sets was increased. Noise was cumulatively added to both data sets (and both the training and test files of each) in steps of 2%. This was achieved by scaling one randomly chosen value in every fifty by a random value. The experiments investigate levels of noise up to 10% (i.e. one in ten values is wrong). Observed levels of noise in the data sets for which this system is designed are around 1%. These experiments were performed with the system using traditional fuzzy logic and non-overlapping membership functions.

Table 7 shows the results obtained from 50 runs of the system for both data sets, at six different levels of noise. Generally, the results show a gradual decrease in accuracy for both data sets. At ten percent noise the accuracy for the Iris data does increase, but this is likely to be chance and the fact that the number of values in the set is insufficient for a 2% noise differential to affect a significant number of data items. However, the accuracy falloff for the larger, Wisconsin Breast Cancer data set is particularly revealing.

Figure 9 shows the rate at which accuracy falls for classification of items in the training and test data sets as noise levels increase. Note the way accuracy falls linearly for the training data, but appears to fall proportionate to the square of the percentage of noise in the test data. This large decrease in performance is likely to be caused by the noise

reducing the homogeneity of the training and test sets, so rules evolved for the training set work less and less well for the test set.

**Table 6:** Mean and best accuracy rates, processing times and intelligibility of solutions when using different clusterers. (Accuracy values in normal intensity indicate results for the training set, bold values show results for the test data set.)

| | Iris data | | | | Cancer data | | | |
|---|---|---|---|---|---|---|---|---|
| **System:** | **Av. Accuracy** | **Best accuracy** | **Av. time** | **Av. # of rules** | **Av. accuracy** | **Best accuracy** | **Av. time** | **Av. # of rules** |
| Using ***basic cluster*** | 97.84%<br>**92.4%** | **93.3%** | 14.9 secs | 1.38 | 97.3%<br>**93.62%** | **95.43%** | 419 secs | 11.3 |
| Using ***k-means*** | 96.67%<br>**95.79%** | **97.3%** | 25.6 secs | 2.94 | 98.6%<br>**94.07%** | **96.0%** | 317 secs | 9.88 |

**Table 7:** Mean and best accuracy rates , processing times and intelligibility of solutions for different noise levels. (Accuracy values in normal intensity indicate results for the training set, bold values show results for the test data set.)

| | Iris data | | | | Cancer data | | | |
|---|---|---|---|---|---|---|---|---|
| **Noise level:** | **Av. accuracy** | **Best accuracy** | **Av. time** | **Av. # of rules** | **Av. accuracy** | **Best accuracy** | **Av. time** | **Av. # of rules** |
| 0% | 96.67%<br>**95.79%** | **97.3%** | 25.6 secs | 2.94 | 98.6%<br>**94.07%** | **96.0%** | 317 secs | 9.88 |
| 2% | 94.67%<br>**97.33%** | **97.33%** | 23.0 secs | 1.0 | 98.51%<br>**93.33%** | **95.71%** | 347 secs | 9.66 |
| 4% | 92.29%<br>**87.47%** | **94.67%** | 18.5 secs | 2.10 | 97.56%<br>**93.87%** | **95.43%** | 380 secs | 8.46 |
| 6% | 74.19%<br>**84.11%** | **94.67%** | 19.3 secs | 1.90 | 96.98%<br>**89.54%** | **92.29%** | 454 secs | 9.82 |
| 8% | 73.95%<br>**84.83%** | **94.67%** | 19.3 secs | 1.82 | 95.67%<br>**86.16%** | **90.57%** | 423 secs | 8.86 |
| 10% | 74.93%<br>**87.55%** | **96.0%** | 19.5 secs | 1.96 | 95.41%<br>**78.02%** | **84.86%** | 523 secs | 13.6 |

Upon consideration, such reduced effectiveness of rules may be manifested in two ways. Firstly rules become 'misled' by the noise (perhaps because of overfitting by too many excessively specific rules) and thus do not generalise well to the test data set. Secondly, the noise disrupts the clusterers, so that the clustering for training and test sets becomes increasingly different. The resulting LOW, MEDIUM and HIGH fuzzy sets for the test and training data become increasing disparate, reducing the effectiveness of the rules further.

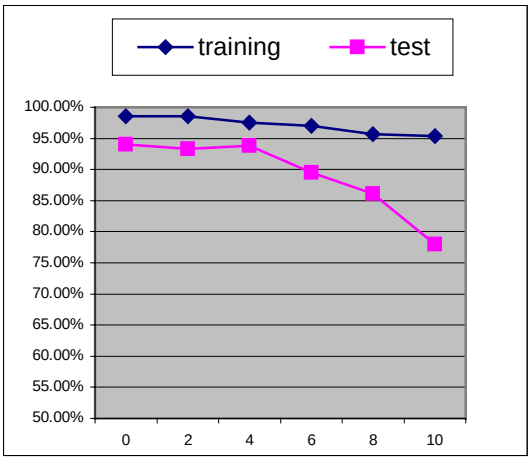


**Figure 9:** Average accuracy for increasing levels of noise in training and test cancer data. These effects are not so obvious for the Iris data where only a linear decrease in accuracy is evident, perhaps because of the small number of data points in this set or because few rules were used, helping rules to generalise without being misled by noise.

Table 7 also shows the number of rules and processing times for different levels of noise. From these results, there does not appear to be any clear correlation between levels of noise in the data and intelligibility or speed.

# 6 Conclusions

This paper has investigated the use of a genetic programming system to evolve fuzzy rules for the purpose of detecting 'suspicious' data amongst 'normal' data. The system contains many novel elements, including a crossover operator designed to minimise disruption, binary genotypes, and a new method for interpreting fuzzy rules designed to preserve all fuzzy set membership values.

Consultation with our collaborating company, Lloyds/TSB resulted in a set of evaluation criteria for the system: intelligibility, speed, handling noise and accuracy.
With these aspects in mind, three sets of experiments were performed on the system, using two standard data sets to permit comparison with the literature. The first test investigated the effect of membership functions on the system. By increasing the overlap between fuzzy membership functions and by preserving the information held in the membership values, the results showed that the number of rules needed to classify data could be reduced. This reduction often led to a decrease in accuracy of classification, but this was offset by the dramatically increased intelligibility of output, faster processing time, and better feature selection. The second test investigated the effects of using different clusterers in the system. It was found that a basic clusterer slightly reduced the accuracy of the system, compared to the more complex K-Means approach. The choice of clusterer did not seem to have any consistent effect on intelligibility of output or

processing speed. The final test investigated the ability of the system to cope with increasing levels of noise in the data. As one would expect, accuracy of classification was detrimentally affected as noise increased. Interestingly, the intelligibility and processing speed showed no clear trend for increasing levels of noise.

Together, these experiments show:

- many factors affect accuracy of classification
- intelligibility and processing speed only seem to be affected by the type and use of membership functions - noise and the choice of clusterer seems unimportant.
- noisy data causes at best a linear drop in accuracy, and at worst, a fall proportionate to the square of input noise.

As with most real-world problems, there is no clearly defined 'best solution' to the problem of detecting fraud by computers. This paper has examined one approach, and has shown that, with the appropriate system components enabled, the use of GP to evolve fuzzy rules can provide intelligible, accurate classification quickly, even for noisy data.

## Future Work

The system will be applied to a set of home insurance data provided by Lloyds/TSB in order to assess its abilities to detect 'suspicious' claims. Since the experiments reported here indicate that different settings of the system provide useful classifications for different data sets, an obvious solution is the use of decision aggregation using a committee approach, allowing the best solution generated by the different models employed by the system to be provided automatically.

## Acknowledgments

Thanks to Hugh Mallinson for his implementation of the clustering algorithms and some of the fitness functions used in the system. Data was provided by the Data Set Repository at the Information and Computer Science Dept., University of California, Irvine. The breast cancer database originated from University of Wisconsin Hospitals, Madison, assembled by Dr. William H. Wolberg:
ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names
Insurance data has been kindly provided by Lloyds/TSB. This project is being performed in collaboration with Searchspace Ltd and Lloyds/TSB, and is funded by the EPSRC, ref: GR/L3/708.

## References

[1] Bentley, P. J. & Wakefield, J. P., 1996, Hierarchical Crossover in Genetic Algorithms. In Proceedings of the 1st On-line Workshop on Soft Computing (WSC1), (pp. 37-42), Nagoya University, Japan.
[2] Bentley, P. J. & Wakefield, J. P., 1997, Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. Chawdhry, P.K.,Roy, R., &

Pant, R.K. (eds) Soft Computing in Engineering Design and Manufacturing. Springer Verlag London Limited, Part 5, 231-240.
[3] Bezdek, J. C. and Pal, S. K. (Ed.s), 1992, *Fuzzy Models for Pattern Recognition*. IEEE Press, New York.
[4] Dasarathy, B.V., 1980, Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 1, 67-71.
[5] Hartigan, J. A, 1975, *Clustering algorithms*. Wiley, NY.
[6] Kosco, B., 1994, *Fuzzy Thinking, the new science of fuzzy logic*. Flamingo. Harper Collins Pub., London.
[7] Koza, J., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.
[8] Koza, J. et al., 1998, *Genetic Programming '98: Proceedings of the Third Annual Genetic Programming Conference*. Morgan Kaufman Pub., CA.
[9] Mallinson, H. and Bentley, P.J., 1999, Evolving Fuzzy Rules for Pattern Classification. In Proc. of the Int. Conf. on Computational Intelligence for Modelling, Control and Automation - CIMCA'99.
[10] Marmelstein, R. E. and Lamont, G. B., 1998, Evolving Compact Decision Rule Sets. In Koza, John R. (ed.). *Late Breaking Papers at the Genetic Programming 1997 Conference*, Omni Press, pp. 144-150.
[11] Mc. Neill, D. and Freiberger, P., 1993, *Fuzzy Logic*. Touch Stone Pub.
[12] Pedrycz, W. (Ed.), 1997, *Fuzzy Evolutionary Computation*. Kluwer Academic Publishers, MA.
[13] Wolberg, W. H., and Mangasarian, O. L., 1990, Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences*, 87,9193--9196.
[14] Yu, T. and Bentley, P., 1998, Methods to Evolve Legal Phenotypes. In Proceedings of the Fifth Int. Conf. on Parallel Problem Solving From Nature. Amsterdam, Sept 27-30, 1998, pp. 280-282.
[15] Zadeh, L. A., 1965, Fuzzy Sets. *Journal of Information and Control*, v8, 338-353.
[16] Zhang, J. (1992). Selecting typical instances in instance-based learning. In *Proceedings of the Ninth International Machine Learning Conference*, pp. 470--479. Aberdeen, Scotland: Morgan Kaufmann.