

## Aberystwyth University

### *Evolutionary approaches to fuzzy modelling for classification*

Shen, Qiang; Galea, Michelle

DOI:

[10.1017/S0269888904000189](https://doi.org/10.1017/S0269888904000189)

Publication date:

2004

Citation for published version (APA):

Shen, Q., & Galea, M. (2004). *Evolutionary approaches to fuzzy modelling for classification*. 27-59.  
<https://doi.org/10.1017/S0269888904000189>

#### **General rights**

Copyright and moral rights for the publications made accessible in the Aberystwyth Research Portal (the Institutional Repository) are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Aberystwyth Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Aberystwyth Research Portal

#### **Take down policy**

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

tel: +44 1970 62 2400  
email: [is@aber.ac.uk](mailto:is@aber.ac.uk)

# Evolutionary approaches to fuzzy modelling for classification

MICHELLE GALEA and QIANG SHEN and JOHN LEVINE

*Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh,  
EH8 9LE, UK*

*email: m.galea@sms.ed.ac.uk, qiangs@inf.ed.ac.uk, johnl@inf.ed.ac.uk*

## **Abstract**

An overview of the application of evolutionary computation to fuzzy knowledge discovery is presented. This is set in one of two contexts: overcoming the knowledge acquisition bottleneck in the development of intelligent reasoning systems, and in the data mining of databases where the aim is the discovery of new knowledge. The different strategies utilising evolutionary algorithms for knowledge acquisition are abstracted from the work reviewed. The simplest strategy runs an evolutionary algorithm once, while the iterative rule learning approach runs several evolutionary algorithms in succession, with the output from each considered a partial solution. Ensembles are formed by combining several classifiers generated by evolutionary techniques, while co-evolution is often used for evolving rule bases and associated membership functions simultaneously. The associated strengths and limitations of these induction strategies are compared and discussed. Ways in which evolutionary techniques have been adapted to satisfy the common evaluation criteria of the induced knowledge – classification accuracy, comprehensibility and novelty value – are also considered. The review concludes by highlighting common limitations of the experimental methodology used and indicating ways of resolving them.

## **1 Introduction**

Evolutionary algorithms have been successfully applied to many search and combinatorial optimisation problems. Their popularity is due in great part to their parallel development and modification of multiple solutions in diverse areas of the solution space, discouraging convergence to a suboptimal solution. Fuzzy rule-based systems, on the other hand, have proven successful in many real-world applications where domain knowledge is imprecise or inexact. A common problem with the implementation of such systems, however, is the acquisition of the production rules on which decision-making is based. Evolutionary algorithms have been extensively applied to this knowledge acquisition task, with the majority of initial work, including reviews of such work, focussing on the tasks of control and function approximation (e.g. Hoffmann, 2001). Interest in this area of research is still very high, as is evidenced by a recent special issue of the journal *Fuzzy Sets and Systems* on

genetic fuzzy systems (Cordon *et al.*, 2004). Articles include new developments and techniques for the learning and evolution of fuzzy classification systems and hierarchical fuzzy models.

This paper examines work addressing the task of classification, presenting an overview of the ways in which evolutionary algorithms have been used for automated fuzzy knowledge acquisition. It does not seek to instruct the reader in the evolutionary techniques mentioned, or in the fundamentals of fuzzy logic-based systems and fuzzy inference. Instead, it illustrates how such techniques have been applied to knowledge acquisition within a supervised learning environment. Where appropriate, an attempt is made to place such work within the wider contexts of specialist machine learning and artificial intelligence subfields, such as inductive logic programming and multi-objective optimisation. It should also be noted that the utility of such work extends beyond this specific task – classification is not only an end in itself but is also an integral part of the process in the operation of intelligent systems designed for automated fault detection, control, and decision-making in general.

The meaning of the term knowledge acquisition in this paper has been broadened to include both automated knowledge acquisition for the development of intelligent reasoning systems, and knowledge discovery from large databases. Much of the work reviewed is within one of these contexts and this then places an added emphasis on inducing knowledge that is comprehensible to the user, i.e. on inducing descriptive fuzzy rules using linguistic variables as opposed to approximate fuzzy rules whose main focus is accuracy.

Section 2 introduces the term classifier systems and makes a distinction between their use for solving reinforcement learning problems and supervised learning problems. Section 3 briefly illustrates how evolutionary computation has been applied in the development of fuzzy rule-based systems in general, while Section 4 discusses the different induction strategies that have been used for knowledge acquisition using evolutionary algorithms. These strategies are in most cases equally applicable to the evolution of crisp rules or decision trees, so that when fuzzy examples are sparse they are supplemented by crisp versions in order to indicate how a solution may be developed.

Section 5 is driven by the increasing importance placed by large organisations on acquiring comprehensible knowledge that may be validated by users, and incorporated into their decision-making processes – it focuses on the common knowledge representations utilised in such cases, that is decision trees and production rules. Section 6 then illustrates how evolutionary algorithms have been adapted to satisfy the common criteria by which the acquired knowledge is evaluated. Section 7 provides a summary of this review, and Section 8 presents general comments on the current state of research effort within the community.

## 2 Classifier systems

Classifier systems were introduced by Holland and Reitmann (1978) and the word 'learning' was added later to give Learning Classifier System (LCS), emphasising that such a system 'learns' how to achieve a particular goal by interacting with its environment and receiving feedback on its actions. This feedback, in terms of a reward, is used to guide the evolutionary development of the system's behaviour, represented as a set of rules each one of which is called a classifier. In the original implementation of classifier systems a complicated credit assignment procedure is used to apportion the reward from the environment between the various rules used in the decision-making, while a genetic algorithm (GA) (Holland, 1975) is used to evolve them.

Within the evolutionary community an LCS is often considered as a particular application of GAs, but debate is rife on this issue especially so within the specialist LCS community. For instance, several researchers argue that an LCS is far more than a GA, that the rule discovery subsystem need not be a GA, that it may be replaced by a different evolutionary approach (Tufts, 1995; Ahluwalia & Bull, 1999), or even by a non-evolutionary approach entirely (Stolzmann, 2001). Others may consider the credit assignment subsystem of an LCS more relevant than the rule discovery component and this then suggests the view of an LCS as primarily a reinforcement learning system. Still others argue that solving reinforcement learning problems is only one application area of learning classifier systems. This question of 'what is an LCS?' is addressed in some detail in (Holland *et al.*, 2000; Kovacs, 2001).

Learning classifier systems have certainly been utilised in the last decade for solving both reinforcement learning (Donnart & Meyer, 1994; Dorigo, 1995), and supervised learning problems (Bonelli & Parodi, 1991; Garrell i Guiu *et al.*, 1998; Holmes, 2000; Bernado-Mansilla & Garrell-Guiu, 2003). In reinforcement learning the specific goal is to solve sequential decision tasks through trial and error interactions with a dynamic environment that provides minimal feedback in the form of occasional rewards and penalties. However, in supervised learning for the purpose of classification, the problem is often greatly simplified. As the work presented in the following sections illustrates, the rule sets generated are composed of rules that do not form action/decision chains and this greatly simplifies the credit assignment procedure for the classifier system. Furthermore, the labelled training data provides considerably more feedback and often constitutes a stable learning environment. This partly explains why many of the examples discussed in this review are simplified versions of the original classifier systems (certainly with respect to the credit assignment procedure), though custom genetic operators tailored for the particular problem domain are often used instead.

Because of this development arising out of the application of classifier systems to supervised learning of non-chained rules, and because of the previously mentioned debate on what actually constitutes a

classifier system, the use of this term beyond this section is generally avoided. For an overview tracing the development of classifier systems in the last decade the reader is directed to (Lanzi & Riolo, 2000), while (Bonarini, 2000) provides a general introduction to learning fuzzy classifier systems and their applications.

### **3 Evolutionary computation and fuzzy modelling**

Evolutionary computation (EC) is the application of methods inspired by Darwinian principles of evolution to computationally difficult problems. Evolutionary algorithms (EAs) re-iteratively apply genetic-inspired operators to a population of solutions, modifying or replacing members of the population so that on average each new generation tends to be better than the previous one, according to some predefined fitness criteria.

EAs have been extensively and successfully applied to combinatorial and search problems. Reasons for their popularity include their broad range of application (e.g. robotics, control, and natural language processing), their relative simplicity of implementation that requires little domain knowledge, and their development of multiple solutions that search different parts of the solution space simultaneously. A more detailed discussion of EC, including strengths and issues, may be found in (Baeck, 1996; Fogel, 1997; Whitley, 2001).

There are several different approaches for reasoning with imperfect or imprecise knowledge (Parsons, 2001), including fuzzy rule-based systems that are based on fuzzy set theory and fuzzy logic (Zadeh, 1965). At the core of a such a system are:

1. A knowledge base composed of fuzzy production (IF-THEN) rules that conceptualise domain knowledge (the rule base – RB), and the membership functions defining the fuzzy sets associated with the linguistic terms employed in the fuzzy rules (the data base – DB);
2. An inference procedure that uses this stored knowledge to formulate a mapping from a given input (e.g. in classification, conditions denoted by attribute values) to an output (e.g. in classification, a conclusion denoted by a class label).

Fuzzy rule-based systems capture and reason with imprecise or inexact knowledge (in fuzzy logic everything is a measure of degree (Zadeh, 1988)), and since many real-world problems contain a measure of imprecision and noise, the application of such approximate reasoning systems in these situations is a viable approach. This is supported by many successful applications in industry and commerce that deal with automated classification, diagnosis, monitoring and control (Hirota, 1993; Bardossy & Duckstein, 1995; Pedrycz, 1996).

Evolutionary techniques have been used to generate the rule base of a fuzzy rule-based system, or fine tune the membership functions, or both. Other applications of evolutionary techniques are related to the pre-processing and post-processing stages of the knowledge discovery process. These include feature construction or selection (Smith & Bull, 2003), training example subset selection (Endou & Zhao, 2002), and rule base optimisation (Nakashima *et al.*, 1998; Ishibuchi & Yamamoto, 2004). Another interesting use is found in (Marin-Blazquez & Shen, 2002) where a GA is used to transform accurate but non-descriptive approximate fuzzy rules into equally accurate descriptive fuzzy rules, while in (Gomez-Skarmeta *et al.*, 2001) different EAs are used to select and tune fuzzy (approximative) rules for classification, from a larger set generated by fuzzy clustering and fuzzy neural networks. For reviews of evolutionary computation techniques that cover some of these ancillary tasks the reader is directed to (Freitas, 2002).

The focus of this review is on rule base generation, but, when the membership functions are simultaneously evolved then relevant work is also discussed. The branches of EC that have been mainly applied to this task are genetic algorithms and genetic programming (GP) (Koza, 1992). Other major branches are evolutionary programming (EP) (Fogel *et al.*, 1966) and evolution strategies (ES) (Rechenberg, 1973). These branches are similar to each other and differ mainly in the representation used for the individuals in a population (e.g. binary strings vs. real-valued vectors), and in the application of the genetic operators such as recombination and mutation of individuals.

### 3.1 Rule base generation

When only the rule base is being induced the membership functions are predefined either by a human expert or some other process (e.g. clustering), and remain fixed throughout the inductive process.

From some of the early work on classifier systems two terms have emerged that are still in common usage today: Michigan-style (Holland & Reitman, 1978) and Pittsburgh-style (Smith, 1980), the names being in recognition of the institutions of the originators of the two approaches. In the first approach one rule is encoded as one individual of the population, whilst in the second later approach a rule base is encoded as one individual of the population. These terms are still used in discussing GAs but within this paper their meaning has been extended to describe the encoding of solutions within the other branches of evolutionary computation where appropriate.

If Michigan-style encoding is adopted then the EA generally evolves either a rule antecedent or an entire rule. In the first case, at each iteration a separate deterministic procedure is used to determine the rule consequent before evaluating the rule for fitness e.g. (Nakashima *et al.*, 1998). Alternatively, the rule consequent, i.e. class, may already be specified for each rule antecedent during the entire run

of EA. This is where an iterative rule learning approach is followed and an EA is run several times in succession with each run concentrating on evolving rule antecedents pertaining to a specific class, e.g. (Gonzalez & Perez, 1998; Romao *et al.*, 2002).

If a complete rule is encoded in an individual then the rule consequent may also be subject to evolution and change by the genetic operators. A restriction may be placed on the crossover operator to ensure that only parents belonging to the same rule consequent are combined to produce offspring (Yuan & Zhuang, 1996; Walter & Mohan, 2000).

With a Pittsburgh-style approach generally both rule antecedents and associated consequents are encoded. The genetic operators may then act on the individual rules within a rule base, or on the composition of the rule base itself. An alternative is proposed in (Yang *et al.*, 2001) where each individual represents a fixed number of rule antecedents. The consequents of the rules are dependent on the number of positive and negative examples they match in the training set and are determined prior to evaluation of the rule set.

### 3.2 Knowledge base generation

The generation of the rule base and optimisation of the membership functions may be done in stages or simultaneously.

If the approach taken is a phased one then generally the rule base is first evolved to an acceptable state using fixed membership functions and afterwards is used to fine tune the original membership functions. An example of this is (Cordon *et al.*, 1998) which is based on *MOGUL* (Gonzalez & Herrera, 1997; Cordon *et al.*, 1999), a methodology for obtaining complete knowledge bases for fuzzy-rule based systems. *MOGUL* suggests three stages: generation of the rule base using predefined fuzzy partitions for the linguistic variables; rule base refinement, through selection of optimal subsets, fine-tuning of individual rules or removal of redundant ones; and finally, a genetic tuning stage that optimises the membership functions.

If the two components of the knowledge base (rule and data bases) are evolved simultaneously, then one of two approaches may be followed. The first approach is to encode both the rule base and the membership functions associated with the fuzzy sets of each fuzzy variable in the same individual (Pena-Reyes & Sipper, 1999; Yuhui *et al.*, 1999). The second approach is that of co-evolution – two populations, one of rules and another of membership functions are evolved simultaneously within a shared fitness environment (Mendes *et al.*, 2001; Pena-Reyes & Sipper, 2001).

## 4 Induction strategies

This section illustrates how EAs have been developed into a solution strategy for automatic fuzzy knowledge acquisition. These strategies (or approaches) are in most cases equally applicable to the evolution of crisp rules or decision trees. However, there are certain differences that arise from the crisp or fuzzy nature of the induced knowledge, and these are highlighted where appropriate.

The various strategies are inter-related (see Figure 1). In the simplest approach an EA evolves a population of rules, rule bases or decision trees, and the end result is the whole of the final population or a subset of it. Note that a path of a decision tree may be considered as an IF-THEN rule, so that a complete decision tree may be considered a rule base. Another approach is iterative in nature, running a basic EA several times in succession with the result of each one being considered a partial solution to the problem. A third approach co-evolves two or more EAs simultaneously, with each EA contributing a component part of the solution. A final approach may in principle use any of the previous strategies, but then combines several different solutions to create an ensemble of classifiers.

**Figure 1** Inter-relationships between induction strategies

Iterative rule learning has generally been used for rule base generation, whilst the basic strategy and co-evolution may be used to simultaneously evolve both the rule base and the data base of a fuzzy rule-based system.

The use of ensembles is perhaps not so much a separate strategy for knowledge acquisition, as a strategy for combining the resulting evolved classifiers. They are however discussed since the problem addressed here is often classification, with generalisation capability therefore being one of the most common and important evaluation criteria (and ensembles of classifiers often provide greater classification accuracy than single classifiers). Furthermore, under certain circumstances there are similarities between the evolution of ensembles and the iterative learning strategy that should be noted (and are discussed in Section 3.4).

### 4.1 Basic induction strategy

As the name suggests this is the simplest way in which an EA may be used to induce rule bases or decision trees. A high-level description of an EA is provided below. Note that the actual genetic operators used and the implementation-level coding of the individuals of a population are dependent on the specific EA being implemented.

Two variants are possible depending on whether an individual represents a partial solution or a



complete solution, i.e. depending on whether an EA uses Michigan-style or Pittsburgh-style encoding for an individual.

Fuzzy systems using Michigan-style encoding to generate rules include *FGA* (Yuan & Zhuang, 1996), *Fuzzy-ROSA* (Slawinski *et al.*, 1999), and the learning systems described in (Ishibuchi *et al.*, 1999b). In these systems an individual represents one rule, has an associated fitness level, and the whole population represents the rule base. A direct consequence is the necessity to maintain a population of different rules that can represent the complete problem domain, since it is unlikely that one rule would be able to explain the entire training set or classify all new instances.

**Figure 2** Basic induction strategy - a generic evolutionary algorithm

In *Fuzzy-ROSA* this issue is resolved by dynamically changing parameter values of the EA. Various indicators of how well the search for rules is progressing are calculated for each generation and, based on these indicators, a separate fuzzy system adapts the genetic operators of the EA. The search indicators include a diversity measure for the individuals of the population based on a heuristic distance measure between two individuals, the best and mean fitness of individuals, and the simplicity or complexity of the rules being generated. For instance, if the average diversity of the population is low then the mutation rate is increased.

Another decision arising from the Michigan-style encoding scheme is whether to utilise the entire final population of the adaptive learning algorithm as the rule base or a subset of the population. The system *FGA*, for instance, uses only a subset of the rules, the aim being to end up with a compact set of high-quality rules. The user defines an accuracy level and all rules that meet that level are selected. The extraction process is then carried out on this smaller population based on three criteria in descending order of importance: accuracy, coverage and fitness (the fitness measure is based on accuracy and coverage of the rule but also on its relative importance within the whole rule set as defined by a uniqueness measure). The rule or rules with the highest accuracy are identified; if there are two or more rules that have an accuracy within a predefined tolerance level then the one/s with the greatest coverage are selected; of these the one with the highest fitness is selected and placed in the final rule set pot. The training examples that are covered by this rule are removed from the training data set and the second rule for the final rule set is selected in the same way. This goes on until the training data set is empty.

In the second variant of the basic evolutionary strategy an individual in the population generally represents an entire rule base and hence only one individual need be selected from the final population

as a solution. Early crisp rule-inducing systems for classification include *GABIL* (De Jong *et al.*, 1993) and *GIL* (Janikow, 1993) while (Harris, 2002) presents a fuzzy extension to *GABIL*. Since each individual is a rule base the search space has consequently increased and the calculation of the fitness function is generally more computationally expensive. However, this encoding does give rise to distinct advantages: the fitness associated with each individual takes into account rule interaction and no additional procedures are required for maintaining diversity in a population.

Ishibuchi *et al.* have conducted an empirical study comparing the two variants of this basic implementation for supervised classification problems. (It should be noted that, in general, such informative comparative studies are rare, whether making comparisons between the variants within one induction strategy, or making comparisons between different induction strategies.) In (Ishibuchi *et al.*, 1996) a Pittsburgh-style individual represents a set of complete fuzzy rule antecedents with each individual having a predetermined number of rules, i.e. all individuals have the same number of antecedents. The rule consequent and confidence factor for each rule antecedent is determined from the training data set by a deterministic procedure. The fitness of this Pittsburgh-style individual is determined by the number of correctly classified training examples.

In the Michigan-style algorithm each rule is represented by a separate individual so that the whole population corresponds to a single rule base. The fitness of an individual here is dependent on the numbers of correctly and incorrectly classified training examples. The authors Ishibuchi *et al.* conclude that for the data sets tested the Michigan-style encoding and resulting algorithm provides a greater classification accuracy at a lower computational cost, when compared with the Pittsburgh-style encoded algorithm.

Ishibuchi *et al.* continue to experiment with these two variants and in later work make more refined observations – the algorithm using Michigan-style encoding for individuals obtains higher classification rates for high-dimensional problems, while the algorithm using Pittsburgh-style encoding obtains higher classification rates on low-dimensional problems (Ishibuchi *et al.*, 2000). This suggests that the Michigan-style encoded algorithm may have a greater ability to find good fuzzy rules in large search spaces. However, this algorithm can not directly optimise a rule base as it only measures the performance of individual rules. The authors suggest that it is this indirect optimisation that leads to inferior results by the Michigan-style algorithm on low-dimensional problems.

Other experiments with the Michigan-style approach (Ishibuchi *et al.*, 1999a) indicate that the classification accuracy deteriorates if the GA is changed from a steady-state to a generational one. That is, if instead of only a few of the worst individuals in a current generation being replaced by fitter offspring in the next generation, all individuals are replaced by offspring. Ishibuchi *et al.* conclude that

in order to maximise performance new rules should be generated from existing good rules (as defined by the fitness function), good rules from previous generations should not die out, and optimisation of a complete rule base should be done directly.

This results in the design of a hybrid algorithm that attempts to combine the best features from both variants. The individuals of this new algorithm are Pittsburgh-style and may have different lengths, i.e. different number of rules. The fitness function used in this hybrid promotes rule bases that are good at classifying the training set but also have a small number of rules. These two elements of the fitness function may be given different emphasis by attaching a weighting factor to each part. The mutation operator is, however, the main change from the original Pittsburgh-style implementation – mutation is now a single iteration of the Michigan-style algorithm and is applied to all generated rule bases after selection and crossover, i.e. the worst rules in each rule base are now replaced with new rules that have been generated from good rules in that same rule base.

In (Ishibuchi *et al.*, 2000) the authors test this hybrid algorithm against their original pure Michigan-style and Pittsburgh-style algorithms. The hybrid achieves the same or better accuracy on all six data sets used, with, in most cases, a smaller number of rules and thereby aiding comprehensibility of the induced knowledge. They also provide a comparison of the CPU time taken by all three variants – the Pittsburgh-style algorithm requires more time than the Michigan-style algorithm, and the hybrid algorithm is even more computationally expensive as it is essentially combining the previous two variants together.

This basic induction strategy may be used to evolve the entire knowledge base of a fuzzy rule-based system, i.e. evolve the rule base and the database simultaneously. The encoding of a Pittsburgh-styled individual may be extended to include the membership functions associated with the fuzzy sets of the fuzzy variables (Pena-Reyes & Sipper, 1999; Yuhui *et al.*, 1999). In (Yuhui *et al.*, 1999), for instance, the GA evolves the individual Mamdani-type rules (Mamdani, 1976) within the rule base, but also the number of rules within the rule base. The fuzzy variables each have the same number of fuzzy sets. This number remains fixed throughout the evolutionary process, but as each membership function is tuned its type (left-triangular, right-triangular, triangular, gaussian, sigmoidal, or reverse sigmoidal) is also allowed to change.

#### 4.2 Iterative rule learning

In this approach a basic EA (as just described in Section 3.1) is run several times in succession, with the output from each iteration considered a partial solution to the problem.

The iterative learning strategy mirrors the separate-and-conquer approach originally applied in

deterministic rule learning algorithms such as *AQ* (Michalski, 1969) and *CN2* (Clark & Niblett, 1989). Here, one rule is built to cover a subset of the training set and then more rules are built to cover the remaining instances recursively. This approach is also termed a covering approach in the literature.

When utilising EAs as the rule discovery mechanism, there are two main variants on the iterative theme – iteration by class where in each iteration rules describing a specific class are learnt, or independent of class where in each iteration good rule antecedents are first found and the class is determined afterwards.

In the induction of crisp rules using the second variant the output of the algorithm may be considered as a decision list, i.e. an ordered list of rules that must be applied in sequence when classifying a new instance (Riquelme *et al.*, 2000). These rules may be considered as a series of IF-THEN-ELSE rules. In the process of classification, if the first rule in the list does not cover the instance, i.e. the corresponding values of the attributes in the rule and instance do not match, then the next one is tried. If the second rule does not work then the third one down the list is tried, and so on. Once an instance is classified by a rule no more rules are tried. If none of the rules cover the instance then a default rule at the bottom of the decision list is used for classification.

**Figure 3** Iterative rule learning strategy

Unordered rule sets, on the other hand, may give rise to conflicts between rules when classifying a new instance, as there may be more than one rule covering the instance and each rule may have a different consequent. For the crisp versions these conflicts are often resolved by a simple majority voting scheme, i.e. the new instance is assigned the same consequent as that of the majority of matching rules. Another common method is to use a distance measure and an instance is then assigned the class of the nearest rule in the rule set (Venturini, 1993; Liu & Kwok, 2000). When classifying by fuzzy rules this overlap between rules is handled by the fuzzy inference method implemented. Of course, there are many possible conflict-resolution strategies, such as recency and complexity-based approaches (Davis & King, 1977; Buchanan & Duda, 1983).

A high-level description for the first variant, i.e. iteration by class, is illustrated in Figure 4.

**Figure 4** Iterative learning - iteration by class

An example system following the iteration by class sub-strategy for inducing crisp IF-THEN rules is *EDRL* (Kwedlo & Kretowski, 1998). At the start of each major iteration, line (1), training examples

with a specific class label are indicated as positive examples whilst all other training examples are indicated as negative. The basic EA of line (4) is a GA using Michigan-style encoding. It is run a number of times, until all or nearly all (as defined by the user) the positive training examples have been covered by a rule, line (3) – each run of the GA outputs the best rule from its final population as determined by a fitness function that encourages larger coverage of positive instances, minimum coverage of the negative ones, and a short rule antecedent. The rule is added to the final rule set and training examples it matches are removed before the next GA is run. Once the inner iteration, line (3), has been completed, the original training set is relabelled with examples belonging to the next class for which rules are to be generated indicated as positive, and all others as negative.

In the case of fuzzy rule induction what constitutes coverage (or matching) of a training instance by a rule needs defining, as an instance may be covered by several different fuzzy rules but to varying degrees. An example of a fuzzy system inducing rules in this manner is *SLAVE*, introduced in (Gonzalez *et al.*, 1994).

A common modification to the class-dependent iterative approach is to simplify the overall algorithm so that instead of running the basic EA several times within a class iteration, it is run only once (see Figure 5).

**Figure 5** Iterative learning - simplified iteration by class

In (Romao *et al.*, 2002) for instance, a GA using Michigan-style encoding is run once for each class with the best individual from each run being added to the final rule set. A main aim here is to obtain a minimum number of fuzzy rules and the resulting rule base compares well in predictive accuracy with other algorithms.

Although evolutionary rule induction systems following an iterative learning approach are generally characterised by utilising a Michigan-style encoding, in (Mendes *et al.*, 2001) a GP algorithm uses a hybrid Michigan-Pittsburgh-style encoding. The individual here represents a *set* of rules, but not a complete rule base. The GP is run once for each class and the best individual from each GP run – here representing a set of disjunctive fuzzy rules for a particular class label – is added to the emerging final rule set.

The second variant of the iterative learning strategy iterates independently of a class label (see Figure 6).

**Figure 6** Iterative learning - iteration independent of class

Reducing the training set between iterations by removing examples covered by generated rules appears to be the most common way of adjusting the training set (line (4)). However, some authors have experimented with interesting alternatives, especially when inducing fuzzy rules. In (Hoffmann, 2004), for instance, an evolution strategy (ES) is repeatedly invoked and each time identifies the fuzzy rule that best classifies the current distribution of training examples. Each training example has an attached weight and Hoffmann employs a mechanism to change the distribution of the examples from one iteration to the next. Examples that have been correctly classified by fuzzy rules generated in earlier iterations have a lower weight, while those that have been misclassified have a higher weight. In each iteration the ES is guided to concentrate on generating fuzzy rules that are best adapted at dealing with the previously misclassified examples.

This algorithm has been applied to the induction of fuzzy rules where each fuzzy rule has its own definition of associated fuzzy sets that are evolved simultaneously with the rules. The resultant fuzzy rules are tested against several other techniques including neural networks and Bayesian classifiers (Michie *et al.*, 1994), and on the whole achieve comparable classification accuracy. However, the algorithm has recently been extended to the induction of descriptive fuzzy knowledge bases for classification and preliminary results indicate that the same classification accuracy is achievable.

An earlier work following a similar approach is that of Junco and Sanchez (2000). The linguistic partition of the input space is defined in advance and a set of descriptive fuzzy rules are generated by a GA. Other differences from the work by Hoffmann lie in the computation of the rule weight and the weight-update scheme for the training examples.

The iterative rule learning strategy therefore works by using different subsets or distributions of the training set, with the aim of extracting just one good rule from each iteration. A benefit arising from this strategy is that it avoids the need for enforcing diversity in the population of the basic EA used.

However, Gonzalez and Perez (1999) highlight the potential shortcomings of this strategy, especially bad when applied to the induction of fuzzy rules. In this work the authors stress the necessity for co-operation between fuzzy rules – since fuzzy rules cover (match) all examples to varying degrees, having a set of co-operative rules is essential to the inference process. This means that it is important to avoid, as far as possible, a situation where an instance requiring classification is matched by two or more rules that have different conclusions. The iterative rule learning approach as it is generally implemented, however, is not particularly conducive to producing co-operative rules since the rule selection process at the end of each iteration does not take into account the previously generated rules,

or the degree of coverage they provide over the entire training examples. In (Gonzalez *et al.*, 1994) an attempt is made to resolve this issue.

In this work (Gonzalez *et al.*, 1994) the authors implement a significant enhancement on their earlier work involving the *SLAVE* system (Gonzalez *et al.*, 1994; Gonzalez, 1995; Gonzalez & Perez, 1998). *SLAVE* follows a class-dependent iterative approach to fuzzy rule induction (as described by Figure 4), with the EA used being a GA. In their later work, however, and in order to encourage co-operation between the induced rules during inference, the authors retain the same iterative approach but do not eliminate training examples between GA runs. Instead, they attach to each example various indicators that are used in the evaluation of a rule when classifying the training examples.

The first indicator gives a measure of the maximum degree of coverage provided by rules in the current rule set having the same class as the example – the maximum positive covering degree of an example. A second indicator gives a measure of the maximum degree of coverage provided by the other rules in current final rule set, i.e. those rules having a different class from the example – the maximum negative covering degree of an example. Depending on the relative values of the maximum positive and maximum negative covering degrees of an example, and the classes of the example and rule being evaluated, the example may be considered as either a positive or a negative one for that particular rule.

The numbers of negative examples and positive examples covered by a rule are used in its evaluation as an indication of its contribution to the completeness and consistency of the current rule set. It should be noted that the values of the indicators are based on the rules already in the final set and these values are therefore modified each time a new rule is added to the final rule set – the new rule may well increase the maximum positive or negative covering degrees of certain examples, and therefore the relative values of the two indicators will also be modified. This dynamic method of keeping track of previously selected rules, and their success at classifying the training set, provides an indication of how the current rule set as a whole acts on the entire training set.

This new version of *SLAVE* is tested against the original, resulting in an improvement on generalisation capability, and, a significant reduction in both the number of rules in the final rule set and the execution time. In this newer version, however, the authors also amend the fitness function, adding in a term to encourage rules with fewer and less complicated conditions in the rule antecedent. Unfortunately, since it is possible that fewer and more general rules (as encouraged by the new fitness function) lead to a final rule set with increased generalisation capability, it is difficult to judge the exact contribution of their new way of adjusting the training set towards the improvements reported.

The enhancement to *SLAVE* is discussed further in (Cordon *et al.*, 1998), as is a different approach to encouraging the generation of a co-operative fuzzy rule base. The alternative proposed is that of adding a post-processing step that works on the generated rule set to further refine the rules and/or remove redundant ones.

### 4.3 *Co-evolution*

From a biological perspective co-evolution is a series of steps during which two or more interacting species undergo reciprocal evolutionary changes – if one species changes the others respond in order to maintain or increase their fitness and this in turn triggers a response in the first species. Models of life that may give rise to co-evolution include predator-prey, host-parasite, symbiosis and mutualism, i.e. variants on the competition or co-operation theme.

From an evolutionary computation perspective co-evolution is distinguished by a fitness evaluation of an individual that takes into account other individuals' fitnesses. In standard evolutionary computation the fitness function is an objective function and outputs the same fitness value for an individual irrespective in which generation it turns up. In co-evolutionary computation the fitness of an individual is also directly related to how well other individuals are doing. This means that strictly speaking it is not necessary to have more than one population to implement a competitive form of co-evolution – in evolutionary computation terms a single population with a shared fitness function also constitutes co-evolution. When more than one population (or species) is present the fitness of an individual is based on its behavior in the context of the other population/s and thus may result in either a co-operative or competitive form of co-evolution.

Co-evolutionary computation is a relatively recent approach to solving optimisation problems with the majority of research being published in the last decade and much of it applied to classification by cellular automata (Wolfram, 1983) or neural networks. The results reported are mixed – for instance, a relative fitness environment might lead to consistently improving individuals in one implementation, but in another might appear to lead to over-specialisation in individuals and degenerate to a cycling between suboptimal solutions (Watson & Pollack, 2001).

The dynamics of co-evolutionary algorithms are more complex than those of conventional evolutionary algorithms, making research on the theoretical underpinnings of the approach both more difficult and essential. Recent work on co-evolutionary computation addresses these challenges – Bucci and Pollack (2002) attempt to provide a mathematical framework for the application of co-evolutionary algorithms, Juille and Pollack (1998) discuss co-evolutionary learning and related issues, while Rosin and Belew (1997), and Potter and De Jong (2000) elaborate on the nature and application



of competitive and co-operative co-evolution respectively.

With regards to utilising a co-evolutionary approach to fuzzy modeling for classification purposes, several examples use a co-operative multi-population model as it seems a natural one to consider when inducing both fuzzy rule bases and associated membership functions. One such example of a specific implementation is *FuzzyCoCo* (*Fuzzy Cooperative Coevolution*, Pena-Reyes & Sipper, 2001). This system co-evolves two populations or species, one of IF-THEN rules and the other of membership functions. Both species are evolved by means of GA. The GA evolving the rules uses Pittsburgh-style encoding, i.e. each individual represents a set of rules. The number of rules for all individuals is predetermined by the user and fixed throughout the GA run. The individuals of the second GA population encode the associated membership functions.

In order to evaluate the fitness of an individual from either species it must first be combined with an individual from the other species. Together they form a complete knowledge base of a fuzzy system and attempt to classify the training examples – the evaluation criteria used by the authors here are the proportion of correctly classified examples and the length of the individual rules within the rule base (the smaller the better). This forming of a knowledge base and subsequent classification of the training examples is repeated a number of times, and the resulting fitness of the individual is either the average or the maximum fitness value obtained from such combinations.

The authors compare the results obtained by *FuzzyCoCo* with those obtained from an earlier work (of theirs) that is evolutionary but not co-evolutionary. In this earlier work an individual encodes both a rule base and associated membership functions. The authors report an increase in the classification accuracy and, when considering the number of fitness evaluations conducted, a decrease in the computation expenditure. *FuzzyCoCo* also scores better on classification accuracy when compared with a system that extracts classification rules from neural networks (Setiono, 2000). The authors believe they have obtained the best results to date on this data set for genetic-fuzzy and neural-network based rule systems.

**Figure 7** Co-evolution of a fuzzy knowledge base – rule base and data base

*CEFR-MINER* (*Co-Evolutionary Fuzzy Rule Miner*, Mendes *et al.*, 2001) is another system that co-evolves two populations using different evolutionary algorithms. A GP algorithm evolves fuzzy trees representing a set of rule antecedents, and an ES algorithm co-evolves the associated membership functions. The fitness value of an ES individual is computed as the sum of the fitness values of a number of individuals from the GP population, where each GP fitness value is based on the number of

correctly and incorrectly classified training examples, when used in conjunction with the membership functions represented by the ES individual.

*CEFR-MINER* is evaluated on several data sets from the UCI data depository<sup>1</sup> and the results are comparable to or better than those obtained by *ESIA* (Liu & Kwok, 2000) and *BGP* (Rouwhorst & Engelbrecht, 2000). However, though both test systems are evolutionary in nature, they generate crisp rules and so in this case it is difficult to establish whether any improvement in classification accuracy is due to the fuzzy nature of *CEFR-MINER* or to its co-evolutionary approach.

Another example of two co-evolving species representing fuzzy rules and membership functions is found in (Casillas *et al.*, 2002). Yet again, the results obtained suggest that co-evolution may provide improved classification abilities when compared with both standard evolutionary and non-evolutionary approaches, though no comparisons on computation efficiency are made.

In (Jeong & Oh, 1999) the rules evolved are of the Takagi-Sugeno type (Takagi & Sugeno, 1985). This example is mentioned as it utilises a multi-population form of co-evolution to generate just the rule base, with each population eventually contributing a component rule. The authors test the rule bases evolved by their co-evolutionary system on a fuzzy control problem under various initial conditions. The results are compared with those obtained by rule bases generated by an evolutionary (but not co-evolutionary) system. The co-evolved fuzzy logic controllers outperform the traditionally evolved controllers on generalisation capability and computational efficiency.

#### 4.4 Ensembles

Ensembles are collections of classifiers whose individual decisions are combined in a certain way when classifying new instances. Note that in learning classifier system terminology a classifier is generally considered to be just one rule. When discussing ensembles in this review, a classifier is considered to be a collection of rules, i.e. a complete rule base.

This combination of individual classifier decisions may make interpretation and validation by human users difficult, but research indicates that very often an ensemble performs better than any of the individual classifiers making it up. For this to be so the classifiers constructed must be different from each other and have uncorrelated errors each less than the error caused by a random classification (Hansen & Salamon, 1990). A more detailed introduction to the use of ensembles within machine learning may be found in (Bauer & Kohavi, 1999; Dietterich, 2000).

Several general methods that may be applied to any learning algorithm for the construction of diverse

---

<sup>1</sup> UCI Repository of Machine Learning Databases, <http://www.ics.uci.edu/~mllearn/MLRepository.html>

ensembles have been developed. These include using different subsets of the training data such as in cross-validated committees (Parmanto *et al.*, 1996) or bagging (Breiman, 1996), using different subsets of the input features (Guerra-Salcedo & Whitley, 1999), and reweighting the training examples after each classifier is created as in boosting (Freund & Schapire, 1996). There are also different ways of combining the decisions of the individual classifiers when classifying a new instance. Common methods include simple majority voting, and weighted voting where each classifier is assigned a weight reflecting how accurate it is considered to be. The weighted classifier will have more or less influence on the final decision depending on its attached weight.

There are two ways in which EAs occur in the creation of ensembles:

1. an EA is used as the learning algorithm constructing the individual classifiers that eventually form an ensemble; or
2. an EA is used to combine classifiers that may or may not have been constructed by an evolutionary algorithm.

Examples of the second use of evolutionary algorithms include (Thompson, 1999; Langdon & Buxton, 2001; Kim *et al.*, 2002; Sirlantzis *et al.*, 2002). However, since the main focus in this review is fuzzy rule base induction using EAs, the discussion concentrates on the first-mentioned use.

Hsu and Hsu (2002) use a class-dependent iterative approach to generate one classifier (see Figure 8 below). The basic EA used in each iteration is a GA. The authors then repeat the iterative process, without making any changes to the training data, to generate several more classifiers whose decisions are combined in a simple majority voting scheme when classifying new instances.

**Figure 8** An approach to generating an ensemble of classifiers

Results of tests on several data sets as reported in (Hsu & Hsu, 2002) indicate that increasing the number of maximum generations of the GA improves the classification accuracy of an individual classifier. Increasing the number of classifiers in an ensemble also increases the classification accuracy of the ensemble. However, the results also suggest that increasing the number of classifiers in an ensemble has a greater positive impact on classification accuracy than increasing the maximum number of generations for an individual classifier.

Yao *et al.* (1996) present two examples of evolved ensembles are presented: the first is an ensemble of neural networks and the second an ensemble of rule-based strategies (for handling the two-player iterated prisoners' dilemma game). A basic EA utilising Pittsburgh-style encoding is used in both

examples: in the first evolutionary programming (EP) is used and each individual is a neural network, and in the second a GA is used and each individual is a rule-based strategy. An ensemble is created by taking a subset of the individuals from the final population.

The methods used for integrating the decisions of the individual classifiers include: simple majority voting, various weighted linear combinations over the whole population, and a weighted linear combination of a subset of individuals from the final population, with the subset being determined by a GA (here, another EA is used to determine the best configuration of classifiers for decision making, i.e. the second-mentioned use of EAs within ensembles). On the whole, the empirical results obtained indicate that ensembles are better at classifying test instances than the single best individual within the final generation.

Several interesting points should be noted about this work. With deterministic learning algorithms diversity must somehow be introduced into the individual classifiers making up the ensemble and this may be accomplished in several ways, as indicated above. Yao *et al.*, however, make no changes to the training data (though this has also been noted for the work by Hsu and Hsu). For their evolved artificial neural networks Yao *et al.* run their basic EP algorithm and use all or a subset of the individuals in the final generation. They use four different combination methods for the resulting non-symbolic classifiers with three different data sets. In nine out of the twelve comparisons the error rate of the ensemble is better (i.e. lower) than that obtained by using the fittest individual from the final generation, and in one other comparison the error rates are the same. This indicates that an EA, which is stochastic in nature, introduces enough diversity into the resulting classifiers such that no manipulation of the training data is necessary.

In inducing their rule-based game-playing strategies Yao *et al.* experiment with creating more diverse classifiers by enforcing speciation within the same population of solutions. Specifically, they implement a fitness-sharing function that discourages individuals from staying at the same high-fitness region. They make comparisons between the best individual from the final generation, the best individual from the final generation where fitness sharing was used, and an ensemble created from the final generation where fitness sharing was used. The ensemble produced the best results. Though it might have been more informative to also compare the results of an ensemble with fitness sharing, against an ensemble of classifiers where no fitness sharing was used (and thereby give a clearer indication as to whether it was the ensemble producing the improved accuracy, and not the relative fitness function), their results suggest that additional diversity between classifiers for an ensemble may be introduced by using current methods within EC for speciation and niching.

Yao *et al.*'s aim in enforcing more diversity between the classifiers in a population is to end up with

individuals that work well on different parts of the problem. This is in the expectation that together these individual classifiers can integrate into a system that successfully handles the complete problem. Note that this is similar in aim to that of the iterative rule learning approach where a rule is learnt that deals with one part of the training set, and then other rules are iteratively learnt that deal with other parts of the training set. Taken together, these rules form a complete classifier capable of dealing with the complete problem.

It might be expected that an individual classifier within an ensemble, being a 'complete' solution (i.e. a rule base), performs better than any and each component rule produced by the iterative rule learning strategy. However, this may not be the case if Yao *et al.*'s approach of enforcing diversity within a population of complete classifiers is taken to the extreme. The only resulting differences might then lie in the size of the component parts of the solution – in Yao *et al.*'s work each classifier may then be considered as a rule set (as opposed to a complete rule base) evolved to handle a particular component of the problem, while in the iterative approach just one rule deals with each component. It would be interesting and informative to compare these two approaches with the aim of understanding their fundamental similarities and differences.

## 5 Knowledge representation

This section provides an overview of the existing literature from the perspective of the representation of induced knowledge. Due to their inherent comprehensibility the focus is placed on work utilising decision trees for their solution, or various forms of IF-THEN rules.

For the interested reader, neural networks are another popular hypothesis language used in evolutionary learning; EAs have been used to define the topology of a neural network, or the weights of a network with a pre-defined topology, or even to define both simultaneously (Yao, 1999). As is commonly recognised neural networks have excellent generalisation capabilities but offer little explanatory advantages to the knowledge expert. This is why a considerable amount of research nowadays is also focused on extracting comprehensible rules from neural networks (Andrews *et al.*, 1995; Mayer *et al.*, 1999).

### 5.1 Decision trees

Decision trees are a popular hypothesis language as they are easy to comprehend and give an explicit model of the decision-making process.

An internal node of an induced decision tree specifies a test on an attribute of the data set (though more complex trees may be built by specifying tests at nodes based on more than one attribute). Each

outgoing branch of the node corresponds to a possible result of the test and leaf nodes represent the class label to be assigned to an instance. To classify an instance a path from the root node of the decision tree is traced to a leaf node; each internal node reached specifies which outgoing branch should be taken, depending on the value of the relevant attribute in the instance. The instance is assigned the class label of the leaf node reached at the end of the path. Each individual path of the tree may also be equated with an IF-THEN rule.

**Figure 9** An example fuzzy decision tree

In traditional machine learning decision trees are induced by following a 'divide-and-conquer' strategy whereby, depending on the splitting criterion used, the training data is partitioned into disjoint subsets and the algorithm is applied recursively to each subset. Examples of such algorithms that induce decision trees are *C4.5* (Quinlan, 1992) and *fuzzy ID3* (Umanol *et al.*, 1994) for crisp and fuzzy decision trees respectively. Such learning algorithms generally use an attribute-value representation as the example language, i.e. each instance in the training data is described by specific values of a given set of attributes. The domain of each attribute may be finite or nominal, or numerical. The numerical attributes are either handled directly by the algorithm or discretised before the induction of crisp decision trees, or they are fuzzified prior to the induction of fuzzy decision trees. Ultimately, such attributes are translated to finite domains.

Within EC decision trees have been used in one of two different ways:

- an EA evolves a population of decision trees, or
- a more traditional decision tree induction algorithm integrates an EA, generally as part of its splitting mechanism for the training data.

In the former approach GP has generally been used to evolve solutions, as it is the tree representation utilised by GP that makes it a natural candidate for the representation of decision trees (Koza, 1991). Work implementing this approach includes (Eggermont, 2002) for fuzzy decision trees, and (Rouwhorst & Engelbrecht, 2000; Zorman *et al.*, 2000; Llorca & Garrell, 2001) for crisp decision trees. An example fuzzy decision tree induced in (Eggermont, 2002) is illustrated in Figure 9.

In (Eggermont, 2002) each continuous attribute is fuzzified by first applying a clustering algorithm on the values, and next a membership function is defined for each cluster. GP is then used to evolve a population of fuzzy decision trees. The function set contains conditions of the form `ATTRIBUTE=VALUE` where value is either a nominal value or a linguistic term with underlying fuzzy semantics, and the terminal set contains the different possible classes. When classifying an instance a fuzzy membership

value for each of the classes is computed and the class with the greatest value is the one assigned to the instance.

This algorithm is tested on several benchmark datasets and the results obtained on misclassification error rates were, on the whole, comparable with those obtained from several other evolutionary and non-evolutionary induction algorithms.

There is little to compare with the above work on evolving fuzzy decision trees. However, more work has been carried out on the evolutionary induction of crisp decision trees. Such work indicates that classification performance achieved by the evolved decision trees is comparable with that of trees constructed by more well-known algorithms such as *C4.5*. However, Rouwhorst and Engelbrecht, (2000), and Zorman *et al.* (2000) also clearly indicate that evolved trees are considerably smaller in size and therefore more comprehensible to humans. Reasons for this are discussed in Section 5.1.

The second way of utilising EAs in the induction of fuzzy decision trees is by creating a hybrid algorithm such as in (Janikow, 1996; Ragot & Anquetil, 2001), or (Cantu-Paz & Kamath, 2003) for crisp decision tree induction.

*2l-FDT* (2-level Fuzzy Decision Tree, Ragot & Anquetil, 2001), for instance, creates a fuzzy decision tree by combining two different fuzzy clustering techniques and a GA. In the first phase partitions of the training data are recursively determined by a fuzzy C-means based clustering algorithm (Bezdek, 1981; Gath & Geva, 1989) integrated with a GA that decides on the best feature subspace for each partitioning. The GA fitness function evaluating each feature subspace is based on a fuzzy adaptation of the information gain measure (Quinlan, 1986). The termination criteria for the recursive partitioning includes the number of training examples in the final partitions, the size of the tree, and the information gain criterion itself. At this point, the second phase, a second fuzzy clustering algorithm as reported in (Krishnapuram & Keller, 1993) is applied on the training examples, one for each class in each of the final partitions or nodes resulting from the previous level; a final leaf is added for each subclass determined. The resulting fuzzy decision tree outperforms the decision tree produced by *C4.5* on both classification accuracy on a test set, and in comprehensibility, i.e. *2l-FDT* produces trees considerably smaller in size and therefore more easily interpretable by human experts.

In (Janikow, 1996) a fuzzy decision tree-building algorithm is augmented with a dynamic optimising procedure for the node partitioning. The basic algorithm is based on *ID3* (Quinlan, 1986) while the optimising procedure is again a GA. After all attributes' values have been fuzzified this algorithm recursively partitions the training data set until a termination criterion is satisfied. Note that the training examples within a node are not partitioned further if they all belong to the same class, or all

attributes have been used in the path leading to the node, or the information content of the node reaches a certain threshold level. Nodes are then processed according to an ordering measure – the node containing the greatest number of examples is processed first, and the partitioning of the training examples is then dependent on selecting the attribute that maximises the information gain measure based on entropy.

Once an attribute has been selected its information content, based on the specific examples in the node, is minimised by modifying its original fuzzy sets. This dynamic redefinition of the fuzzy sets of an attribute softens the constraint made by *ID3* and its fuzzy adaptations, that real-valued attributes must be partitioned (or fuzzified) just once prior to tree-building – it is aimed at increasing the consistency of the fuzzy tree. It should be noted that in order to preserve the comprehensibility of the induced tree, at the possible expense of generalisation capabilities, the number of fuzzy sets of an attribute remains fixed during tree-building and the optimisation procedure is carried out only once.

In the crisp decision tree induction example cited above (Cantu-Paz & Kamath, 2003), the authors experiment with the induction of oblique decision trees. Many of the standard decision tree induction algorithms are axis-parallel algorithms: the tests at each node involve only a single attribute which makes the test equivalent to a hyperplane parallel to one of the axes in the attribute space. Oblique decision tree induction algorithms use multivariate tests, i.e. tests based on more than one attribute and that are not necessarily parallel to an axis. The final tree may be smaller and more accurate, but this is sometimes at the expense of comprehensibility, since the multivariate test at each node is more difficult to interpret. In this work Cantu-Paz and Kamath adopt a similar approach of integrating an evolutionary algorithm as the splitting mechanism of the oblique decision tree induction algorithm *OCI* (Murthy *et al.*, 1994). They produce two hybrids: *OCI-GA*, *OCI-ES*, where GA is a genetic algorithm and ES an evolutionary strategy. These hybrids are tested against the original *OCI* and three other axis-parallel and oblique decision tree induction algorithms on several artificial and benchmark datasets. The performance criteria are accuracy, tree size and execution time. The conclusions drawn by the authors are that the evolutionary hybrids are capable of finding oblique trees with similar or higher performance measures than the other algorithms, and the research therefore merits further investigation.

## 5.2 Production rules

The majority of the work in knowledge discovery via evolutionary computation has used various restricted forms of first-order logic as a hypothesis language. Much of this has been in the form of simple propositional IF-THEN rules, i.e. a conjunction of crisp or fuzzy conditions leading to a crisp or fuzzy conclusion. These are generally equivalent to the rules that may be extracted from decision



trees.

Slightly more expressive are propositional rules with internal disjunction between attribute values. Work has also been carried out in encoding into solutions disjunctions between attributes and negations of attribute values. The example language, i.e. the training data language, is generally an attribute-value representation. The rules induced may or may not include a confidence measure that is used in fuzzy reasoning when classifying a new instance. Examples of work inducing such fuzzy rules include (Ishibuchi *et al.*, 1995) – simple propositional, (Yuan & Zhuang, 1996) – internal disjunction, and (Mendes *et al.*, 2001) – disjunction between attributes and negation. These rules may take the following form:

$$\begin{aligned} \text{Rule } R_j: & \text{ IF } A_1 \text{ is } (v_{11} \text{ OR } v_{12}) \text{ AND } A_2 \text{ is } v_{21} \text{ AND } \dots \text{ AND } A_n \text{ is NOT}(v_{n1}) \\ & \text{ THEN } \textit{Class is } C_j \text{ with } CF=CF_j \end{aligned}$$

where  $A_1$  to  $A_n$  are the attributes in a data set,  $v_{ik}$  is a specific linguistic term of attribute  $A_i$ ,  $C_j$  is the rule consequent of rule  $R_j$ , and  $CF_j$  is the rule confidence factor.

Depending on the encoding and genetic operators used within an EA incomplete fuzzy rules may be induced, i.e. not all attributes need be present in the rule antecedent leading to shorter rules that may also therefore be more comprehensible. For instance, in (Yuan & Zhuang, 1996) where a particular dataset has four attributes (OUTLOOK, TEMPERATURE, HUMIDITY and WIND), rules may be generated that do not include all four attributes, such as:

$$\text{IF (OUTLOOK is Sunny OR Cloudy) AND TEMPERATURE is Hot THEN } \textit{Swimming}$$

The particular subfield of machine learning that is concerned with learning more complex knowledge from training data than can be represented by propositional logic, is that of Inductive Logic Programming (ILP) (Muggleton & De Raedt, 1994). This type of learning is often termed relational learning as the learned description may specify relations between parts of an object. ILP is distinguished by the use of background knowledge (in addition to the use of concrete examples), that acts as a model or template for the solution that must be learned by the algorithm. This background knowledge therefore restricts the solution search space and makes the computation more tractable.

ILP was originally focused on program synthesis in logic languages such as Prolog but such programs may be considered as rules and a consequent shift has occurred in research that also emphasises the use of ILP methods for knowledge discovery in databases. Here, a distinction is often made between *predictive ILP* where the aim is the learning of classification and prediction rules, and *descriptive ILP* where the aim is the learning of clausal theories and association rules. This review concentrates on the

former aim of ILP but the reader is directed to (Lavrač, 1998) for a more general but detailed discussion of ILP status, issues and future trends. One of the challenges identified here is the development of inductive logic programming algorithms for first-order fuzzy systems. For specific examples of fuzzy logic programming environments such as fuzzy extensions to Prolog, the reader is directed to (Baldwin *et al.*, 1995; Munakata, 1998; Ebrahim, 2001).

For first-order classification rule induction the examples in the training data are commonly described by ground facts, i.e. logical formulas that contain no variables and have exactly one predicate that is positive. The hypothesis language is frequently a restriction of Horn clauses, i.e. clauses containing at most one literal. Well known non-evolutionary non-fuzzy algorithms include *FOIL* (Quinlan, 1990), *GOLEM* (Muggleton & Feng, 1990) and *PROGOL* (Muggleton, 1995) which are mainly concerned with the learning of single predicates from positive and negative examples and background knowledge. Fuzzy non-evolutionary ILP learners include (Martienne & Quafafou, 1998; Leung *et al.*, 1999; Shibata *et al.*, 1999; Drobnic *et al.*, 2003). The first three examples are fuzzy extensions of the aforementioned *FOIL* algorithm while the fourth example combines rough sets with fuzzy sets for the induction of fuzzy relational descriptions. Evolutionary non-fuzzy ILP algorithms, on the other hand, include *REGAL* (Giordana & Saitta, 1993), *GLPS* (Wong & Leung, 1995a), *SIA01* (Augier *et al.*, 1995), *DOGMA* (Hekanaho, 1998), and *ECL* (Divina & Marchiori, 2002).

There are far fewer examples of evolutionary fuzzy algorithms that induce logic programs. However, with the observed convergence of learning paradigms it is expected that more work will be carried out in this area. One example is *LOGENPRO* (Wong & Leung, 1995b), a generalisation and extension of *GLPS* mentioned previously. *LOGENPRO* aims to combine the expressiveness of inductive logic programming with the global exploration strength of GP in order to evolve logic programs. Due to the fact that in principle a program may be represented as a parse tree, the authors emphasise that the system is flexible enough to generate programs in different programming languages including Fuzzy Prolog as formulated by Li and Liu (1990).

**Figure 10** Evolving Fuzzy Prolog programs by *LOGENPRO*

In *LOGENPRO* populations of viable programs are evolved to increasing fitness levels. This viability is ensured by the use of a logic grammar specific to the problem and target language of the hypothesis, and is applied in the generation of the initial population and in the application of the genetic operations. The initial population may be randomly generated, induced by other learning systems or provided by the user. The termination criterion is met if either the maximum number of generations have been created, or a program with the required fitness has been evolved. The authors have also

devised specialist crossover and mutation operators. In their Fuzzy Prolog program induction example *LOGENPRO* is provided with positive and negative examples of the target fuzzy relation `can-reach(X,Y)`, background knowledge in the form of the relation `linked-to(X,Y)`, and a logic grammar that stipulates correct syntax of induced programs.

**Figure 11** A simplified example fuzzy program found by *LOGENPRO*

*LOGENPRO* was tested on several problems and against *FOIL*. The authors conclude that it is a promising alternative and in some cases superior when inducing from imperfect and noisy examples. In later work (Wong, 2001), the authors develop another system called *GGP* (Generic Genetic Programming) that combines ILP and GP to induce knowledge from databases. Among the representations possible for the induced knowledge are fuzzy petri nets (Chen *et al.*, 1990) that in turn may be used to represent fuzzy production rules of a knowledge base. The best induced fuzzy petri net is tested and its classification accuracy is found to be slightly better than that obtained by *C4.5* on the same data set.

## 6 Evaluation criteria

A main requirement nowadays in automated knowledge acquisition is that the acquired knowledge should be comprehensible to human users. Two elements that help meet this requirement are the knowledge representation used (discussed in the preceding section), and the simplicity or size of the induced knowledge (Section 5.1). Intuitively, the more concise the acquired knowledge, the easier it is for humans to comprehend and validate. In addition to this, for the development of intelligent reasoning systems classification is generally the core problem, and hence good generalisation capabilities for the induced knowledge are essential (Section 5.2). Within data mining, however, the discovery of novel or interesting knowledge may be considered just as important, or more so (Section 5.3).

It should be noted that very often, authors impose more than one evaluation criterion on their induced classifiers, effectively turning the problem into a multi-objective optimisation one, minimising a set of objectives subject to some constraints. A common method for selecting or evaluating individuals in a population based on multiple criteria, is that of using a fitness function that includes an element for each of the criteria, e.g. a three-objective aggregated fitness function weighted according to user prioritisation of the evaluation criteria:

$$fitness = w_1 (simplicity) + w_2 (accuracy) + w_3 (novelty)$$

More sophisticated techniques for multi-objective optimisation have been developed. One such method is based on the concept of a Pareto optimal solution set (Pareto, 1896), where the user is provided with a set of potential solutions, instead of just one, with each solution offering a different tradeoff between the set of objectives (evaluation criteria) used. This method has recently been applied to evolutionary fuzzy modelling and two examples (for function approximation) are (Gomez-Skarmeta *et al.*, 1998; Jimenez *et al.*, 2001). An example of work utilising this technique for evolving (non-fuzzy) classifiers is (Llora *et al.*, 2002), where the objectives are to minimise both the accuracy of the classifier and the number of rules comprising the classifier (see Figure 12 below, adapted from Llora *et al.*, 2002).

**Figure 12** Set of potential solutions (classifiers) rated according to two criteria - complexity and accuracy

Other evolutionary techniques for solving multi-objective optimisation problems are based on fitness sharing, niching and imposing mating restrictions. Such research is very active and several reviews already exist, including (Coello Coello, 1999; Van Veldhuizen & Lamont, 2000). The following subsections will focus on issues pertaining to the individual evaluation criteria, and not on multi-objective optimisation techniques.

### 6.1 *Simplicity of acquired knowledge*

A particularly advantageous feature of decision trees and rule sets induced by evolutionary-based induction strategies appears to be their simplicity, i.e. their relative small size when compared with decision trees and rule bases produced by many deterministic learning algorithms. This simplicity may result in computational advantages (for instance, simpler rule sets require both less storage space and less computation), and promotes the interpretability of the induced knowledge. Furthermore, this simplicity of the induced knowledge coupled with a fuzzy linguistic representation makes the output particularly comprehensible. Two aspects relating to the size of the induced knowledge are relevant – the number of rules in a rule base or branches in a decision tree, and the number of conditions in each individual rule or branch – the smaller the better in both cases.

A pre-processing step such as feature subset selection helps in generating rules and branches with a fewer number of conditions, whilst a post-processing step such as the removal of redundant rules or pruning of branches from decision trees helps to keep the rule base or tree compact. Redundant rules include duplicate rules and more specific rules that are covered by more general ones within the rule base. For instance, *FGA* (Yuan & Zhuang, 1996) and *SIA* (Venturini, 1993) each have a procedure (based on different criteria) that extracts from the population the minimum number of rules required to

formulate the final rule base.

Other measures to promote simplicity are integrated within the induction strategy. Many of these are for encouraging general rules that cover as many positive training instances as possible (which in turn may lead to rule bases with a smaller number of rules). This necessitates the generation of incomplete rules i.e. rules that do not have a value for each antecedent attribute within the data set. Or, rules with attributes that are allowed to have more than one value, cojoined by a disjunctive operator. When using a Michigan-style encoding, for instance, a rule may be encoded by a fixed-length bit string. This string will have several different segments, one for each of the different predictor and conclusion attributes. Each segment (or gene) has a fixed number of bits with each bit representing one particular value from the domain of that particular attribute. If a bit is turned on then it means that the parent attribute takes that particular value.

When the genetic operators are applied to such a string the result may be an offspring or mutated string that has:

- more than one bit turned on for a particular attribute, and may therefore be interpreted as an attribute with internal disjunction, e.g. TEMPERATURE IS MILD OR HOT; or,
- all bits may be turned on (or off) and that attribute may then be considered as not relevant for that particular rule.

If the Michigan-encoded individuals within a population are not restricted to being represented by fixed-length strings, then operators that encourage shorter rule antecedents may be implemented. A custom mutation operator, for instance, may simply delete the gene representing a specific attribute, whilst a recombination operator may combine two parents of same or different lengths and produce offspring of same, bigger or smaller lengths. Similar operators may be used on Pittsburgh-encoded individuals that represent a complete rule base. Here, a mutation operator may delete an entire rule, whilst a recombination operator may lead to individuals with different numbers of rules.

Yet a different approach to controlling the number of rules within a rule base is found in (Yuhui *et al.*, 1999). Here, a GA evolves a population of Pittsburgh-styled individuals (fixed-length integer strings), where each represents a rule base and associated membership functions. The fuzzy variables each have the same number of fuzzy sets, which is fixed throughout the evolutionary process, but the membership function type (e.g. triangular and sigmoidal) is allowed to evolve. The fixed-length string therefore imposes a maximum number of rules for the individual. However, the genetic operators evolve both the individual rules and the number of rules within the rule base. The latter is accomplished by evolving one particular bit in the string representation of an individual – this bit

indicates the number of rules that should be considered when evaluating the individual on the training set.

As stated earlier, the fitness function is another measure often used to encourage shorter rules or rule bases. When evolving fuzzy rule-based classifiers, classification performance on the training set is obviously an important measure of how fit an individual is, but an additional element may be added into the fitness function that takes into consideration simplicity. With respect to Michigan-styled individuals this additional measure often takes into account the number of conditions in the rule antecedent, while in Pittsburgh-styled individuals both the number of rules within the individual, and the total number of conditions in all rules of the individual may be considered. Each element may be weighted differently, according to user determination of the relative importance of classification accuracy and model comprehensibility. Issues related to producing more general rules within a fuzzy evolutionary environment, and ways of resolving them, are further discussed in (Carse & Pipe, 2002).

## 6.2 Generalisation capability

Generalisation in this context is the predictive capability of an induced classifier on previously unseen test examples. With regards to this property classifiers induced by evolutionary-based methods are typically at least comparable with those obtained from more well-established traditional methods.

The representation utilised by a learning algorithm for the training examples and induced knowledge may impact on the quality of the resulting classifier. In particular, if the knowledge representation language is limited and eliminates the models (decision trees or rule bases) from the search space, then any classifier produced will have poor generalisation capabilities.

Other steps taken to encourage good generalisation are often aimed at avoiding over-fitting of the training data. These same steps may also result in simpler classifiers – for instance, pruning of an evolved decision tree may improve both its simplicity and generalisation capability. In (Llora *et al.*, 2002), for instance, Pittsburgh-style encoded evolutionary learning systems are developed, and the problem of selecting a high accuracy model that also has good generalisation power on unseen test cases is translated into a two-objective scenario that evaluates a model on both its accuracy and complexity (i.e. size).

Although it may be intuitively appealing to believe that parsimonious models always encourage greater generalisation capabilities, evidence to the contrary does exist. In (Cavaretta & Chellapilla, 1999) the authors' goal is to generate models with high generalisation capability, using an algorithm with no predefined limit or bias on the size of the generated model. They use GP to design two algorithms identical in all respects except for the fitness function – the second algorithm has an added

component in the fitness function that takes into account the complexity of each individual. Individuals with a comparable accuracy on the training set but a lower complexity get a higher fitness value. The two algorithms are tested on a binary classification problem. The results indicate that the no-complexity-bias algorithm evolves models with greater generalisation power. Furthermore, analysing only the models generated by the low-complexity-bias algorithm, the more powerful generalisation models had a higher complexity. The authors conclude that their results support their hypothesis that for this problem at least, there is little or no relationship between small model size (as measured by the number of nodes and leaves) and greater generalisation powers.

Specifically for GP, Bersano-Begey and Daida (1997), and Kushchu (2002) present reviews of techniques that have been used to promote generalisation. These include a fitness function that also encourages simplicity of the solution (the premise is that a more general solution is less likely to over-fit the training data), inserting noise in the training data (again discourages over-fitting), evaluating solutions on different subsets of the training data in different generations (penalises over-specific individuals which rely on a specific subset of the training examples for their fitness value), and co-evolution (as discussed in Section 3.3). Furthermore, as indicated in Section 3.4, an ensemble of classifiers may result in greater classification accuracy than the single best classifier in the group. Keijzer and Babovic (2000) for instance, provide an argument based on both theoretical considerations and experimental evidence, that the use of ensembles of GP-created solutions can improve generalisation capability (tested on symbolic regression problems). This is attributed to the reduction in error due to variance.

Kushchu (2000) tests the use of problem-specific functions versus more general functions in the creation of GP trees. The reported results suggest that the use of more general functions may evolve solutions with better generalisation capabilities, at least for the relational problem tested (n parity problem). This is attributed to the fact that using specialised non-terminals for the construction of a GP solution may increase the representational bias too much. That is, the use of more general non-terminals may mean that there are different ways of representing the same optimal solution and therefore the evolutionary search process is more likely to find one of them.

A major problem, however, is that no specific guidelines exist on how to promote generalisation – though there is work supporting each of the methods presented above, there is also work providing evidence to the contrary, i.e. the use of these methods do not always lead to solutions with greater generalisation capabilities.

Yet another concern is the testing methodology. Rowland (2003) attempts to clarify some of the issues surrounding model selection and validation in supervised learning via evolutionary computation, but

Kushchu (2002) emphasises that the experimental methodology used is not always consistent with testing for generalisation, and stresses the need for separate training and test sets and for following proper procedures in determining these sets.

### 6.3 Novelty value

Knowledge discovery from data bases involves mining large volumes of data for new knowledge that may be utilised in the operational, managerial and strategic planning processes of an organisation. In this context the novelty aspect of the acquired knowledge, i.e. how surprising or interesting it is to the user, may be as important as its generalisation capability.

What constitutes 'interesting' or 'surprising' knowledge in general is open to debate. Objective measurements based on information theory or statistics have been used, see (Freitas, 1998; Hilderman & Hamilton, 1999; Suzuki & Kodratoff,) for instance. More subjective measurements may also be considered that are generally based on the views of users within an organisation making use of the acquired knowledge (Silberschatz & Tuzhilin, 1996; Liu *et al.*, 1997). Here, the general premise is that acquired knowledge that confirms a user's beliefs or suspicions, useful as that may be in itself, is not novel, interesting, or surprising to the user.

Attempts to induce interesting knowledge using evolutionary computation have been made, but this is still a relatively unexplored area. In (Romao *et al.*, 2002) fuzzy IF-THEN rules are induced where the subjective degree of interestingness of a rule is based on impressions of what a user expects the relationship between a set of predictor attributes and a conclusion attribute to be. These user impressions are stipulated in the same form of IF-THEN rules, and are defined prior to running the GA, and then kept fixed during the rule discovery process. At each iteration each rule is compared with each of the predefined user impressions (see Figure 13 below).

#### **Figure 13** Matching between rules and user impressions

The larger the degree of similarity between the IF parts of a rule and a user impression respectively, and the larger the degree of contradiction between the THEN parts, then the greater the degree of surprise to the user. The final degree of surprise for a rule is the maximum degree obtained between the rule and a user impression. When evaluating the fitness of the rule this degree of surprise is used by the fitness function, which also takes into consideration the predictive accuracy of the rule over the training set.

Freitas (1999) incorporates a more objective measurement for the degree of interestingness into the



fitness function. In this work the algorithm *GA-Nuggets* evolves a population of crisp IF-THEN rules where the genetic operators are applied to the rule antecedent, but the rule consequent is determined on the basis of goal-attribute values in the training set. The training set may have more than one goal attribute, with each goal attribute having its own set of goal values. However, an evolved rule is only allowed to have one rule consequent chosen from the values of one of the goal attributes. The degree of interestingness is based on this chosen goal-attribute value.

The rationale behind this measurement of interestingness is as follows. Let  $G_i$  denote a given goal attribute and  $g_i$  denote a specific value within its domain. The greater the relative frequency of  $g_i$  in the training set, the easier it is to discover a rule predicting its value, and consequently, the less interesting the rule may be considered to be. Accurate rules predicting a more rare goal attribute value are therefore assigned a higher fitness value. Similar work utilising an objective measurement for rule interestingness in the fitness function is (Noda *et al.*, 1999; Noda *et al.*, 2002). Here the main aim is the discovery of interesting rules after the user specifies the goal attributes of particular interest.

## 7 Summary

This paper has presented an overview of the possibilities of applying evolutionary computation to fuzzy modelling, and illustrated how this may be achieved with respect to rule base and knowledge base generation. From these discussions it is clear that a low-level dependency exists between what an individual represents, and the number of individuals selected from the final population. It is this inter-dependency that gives rise to the various approaches.

**Figure 14** Inter-dependency between induction strategy, representation of individuals, and number of individuals selected

For instance, consider the case where an individual represents one rule. If the entire final population or a subset of the final population is selected then that leads to one variant of the basic EA (utilising Michigan-style encoding). Alternatively, if just one individual from the final population is selected then it will be necessary to rerun the basic EA until a set of rules covering the complete training set is generated, thereby leading to the iterative rule learning approach. If, however, an individual represents a rule base, then selecting the best of the final population as the final solution gives rise to the second variant of the basic EA (utilising Pittsburgh-style encoding). Alternatively, selecting a subset or the whole population results in an ensemble of classifiers.

In rule base generation the membership functions are typically predefined. The rules may be evolved simultaneously using a basic EA. If an individual in the population represents one rule then diversity

in the population must be enforced to ensure a reasonable selection of different rules covering different parts of the problem domain. If the aim is to evolve chained action/decision rules, then the fitness calculation is especially complex and computationally expensive. If an individual represents an entire rule base then the search space is increased significantly, though work exists that suggests this approach produces rule bases with better classification abilities for low-dimensional problems.

This paper has illustrated that the iterative learning approach attempts to simultaneously resolve a few of the issues arising from the two variants of the basic EA implementation. However, since this approach induces rules sequentially, and possibly on different parts of the training set, additional measures may be required to encourage co-operation between the fuzzy rules generated. With regards to co-evolution, utilising multiple populations where each population focuses on evolving rules describing a particular component of the problem, might alleviate the need for enforcing diversity within the individual populations, and for additional mechanisms to encourage co-operation between rules from different populations in the inference process. However, the theoretical foundations of co-evolutionary computation are still in the very early stages of being established and mixed results are not uncommon.

Ensembles comprising rule bases generated by evolutionary techniques may provide better classification accuracy and generalisation capabilities, but evidence in this area is sparse and the way a decision is reached is not always particularly comprehensible to human experts desiring to validate the induced classifiers.

This review has indicated that with regards to knowledge base generation there are three main approaches. The first is a phased method: using predefined membership functions a rule base is evolved, which in turn is used to fine tune the original membership functions. The second approach is one variant of the basic induction strategy, where both a rule base and the membership functions are encoded in the same individual. The third approach is that of co-evolution where two different populations, one of rule bases and the other of membership functions, are evolved simultaneously. Similar issues as for rule base generation hold. Although the major issues associated with each approach are known, little work has so far been directed at empirically or analytically comparing these approaches.

This paper has also discussed the common representation schemes used for evolving knowledge that is comprehensible to human users. Presented with ever more complex data generated by both the scientific and commercial communities, more research is being expanded on learning algorithms that use a rich representation capable of adequately describing the underlying knowledge whilst maintaining human comprehensibility. This is evidenced by the active Inductive Logic Programming

research, increasingly utilising both traditional relational learning techniques and newer ones incorporating evolutionary algorithms.

Furthermore, the common evaluation criteria used for induced knowledge in the context of developing intelligent reasoning systems and data mining have been addressed. Very often more than one (possibly conflicting) evaluation criterion may be adopted and the problem is then a multi-objective optimisation one. In general, there are several ways of encouraging an evolutionary-based induction strategy to produce rule bases or decisions trees that satisfy the individual criteria. These methods may be as part of a pre- or post-processing procedure, i.e. before or after a rule base or decision tree has been evolved; or arise out of the specific encoding used for an individual and of the genetic operators used on them; or as part of the evaluation and selection process of an individual.

## **8 Conclusion**

Much research effort has been directed at applying EC to fuzzy modelling, and yet it is difficult to clearly state and justify how successful it has been. This is due mainly to the experimental methodology employed, inadequate reporting of the pertinent facts necessary for a faithful reproduction of the work and results presented, and an insufficient interpretation of the results that may provide an explanation for the difference in performance. For instance, several papers reviewed report only training error, and not generalisation capability. In other cases it is difficult to identify exactly what comparisons are being made – if a fuzzy evolutionary induction algorithm is compared against a non-fuzzy deterministic induction algorithm, for example, to what should an improvement in accuracy be attributed? Is it to the evolutionary-based learning approach, or to the fuzzy set-based representation language and inference mechanism? As for comparisons of efficiency, when CPU times are compared how much is a performance improvement due to a different programming environment and individual style or skill?

Other differences in performance criteria might arise from the testing methods employed (e.g. random 70:30 split of a data set for training and testing of one algorithm, versus 10-fold cross-validation testing of the same data set by another algorithm), use of different de/fuzzification processes and inference methods for classification, and whether instances with missing attribute values are included or excluded from the training set. In short, it is not apparent that all authors ensure similar testing conditions and environments when evaluating their new or amended induction algorithms. In such circumstances it is difficult to make fair comparisons and draw conclusions with any reasonable degrees of certainty.

However, some of these limitations may not be difficult to resolve, such as making the necessary

information easily accessible to all. Researchers appear to use a common pool of data sets on which to test their algorithms, mostly available from the UCI machine learning depository. It is conceivable that this data base, or one like it, could be extended to hold up-to-date details on classification accuracy achieved by various learning approaches, for instance, with details of the testing methods used to obtain them, or where to obtain such additional information. Of course, this would necessitate researchers to forward the relevant information about their work, and an agent to update the data base (or provide the facilities for researchers to update it themselves). However, it would in turn make it a quick and simple matter for researchers to obtain the information necessary to ensure a valid comparison of their work with that of others, thereby adding credence to new results reported. Such a proposition may be feasible in this age of constant and easy communication technologies.

The more serious limitations relate to the experimental methodology used, and to the interpretation and justification of the results presented. With regards to the former, research in performance measures and statistics that should be considered when comparing learning algorithms appears to be increasing (Alpaydin, 2001; Ling *et al.*, 2003; Nadeau & Bengio, 2003), while other authors aim to provide guidelines on experiment design and methodology in general, including aspects such as the programming environment, the experimental data that should be collated, and how the data may be analysed (Gent *et al.*, 1997; Kainz & Kaindl, 1998; Langley, 2000). In providing a possible justification of the reported results, a main drawback may be a lack of firm or sufficient theoretical foundations for some of the EC branches. It appears this limitation may also be in the process of being addressed. This is evidenced by the organisations and research programmes that have been initiated in recent years to look into the theoretical foundations and potential applications of evolutionary computation in particular (e.g. EvoNet<sup>2</sup>), or into complex adaptive systems in general, utilising technologies such as fuzzy systems, neural networks, evolutionary computation and swarm intelligence (Bonabeau *et al.*, 1999) (e.g. EUNITE<sup>3</sup> and the Santa Fe Institute<sup>4</sup>).

The application of EC to fuzzy modelling is a very active research area drawing people from different communities including machine learning, data mining, and fuzzy systems. Each community has its own individual perspective and priorities, but it is conceivable that any one may learn from the failures and successes of the others. The results reported in the literature also suggest that the use of EC for fuzzy modelling may be a promising area of research. However, for any real progress to be made, and for the sake of credibility of such work, it may now be imperative to implement acceptable experimental research and reporting methods, and just as importantly, to consolidate the available

---

2 EvoNet – European Network of Excellence in Evolutionary Computing, <http://evonet.dcs.napier.ac.uk>

3 EUNITE – European Network on Intelligent Technologies for Smart Adaptive Systems, <http://www.eunite.org>

4 Santa Fe Institute, <http://www.santafe.edu>

information and experience.

## References

- Ahluwalia M, and Bull L, 1999. "A genetic programming-based classifier system" *Proceedings of the Genetic and Evolutionary Computation Conference* pp 11-18.
- Alpaydin E, 2001. "Assessing and comparing classification algorithms" *International Symposium on Adaptive Systems*.
- Andrews R, Diederich J, and Tickle AB, 1995. "A survey and critique of techniques for extracting rules from trained artificial neural networks" *Knowledge-Based Systems* 8(6) pp 373-389.
- Augier S, Venturini G, and Kodratoff Y, 1995. "Learning first order logic rules with a genetic algorithm" *The 1st International Conference on Knowledge Discovery and Data Mining* pp 21-26.
- Baack T, 1996. *Evolutionary Algorithms in Theory and Practice*, Oxford: Oxford University Press.
- Baldwin JF, Martin TP, and Pilsworth BW, 1995. *FRIL – Fuzzy and Evidential Reasoning in Artificial Intelligence*, Research Studies Press, John Wiley.
- Bardossy L, and Duckstein L, 1995. *Fuzzy Rule-based Modeling with Application to Geophysical, Biological and Engineering Systems*, CRC Press.
- Bauer E, and Kohavi R, 1999. "An empirical comparison of voting classification algorithms: bagging, boosting, and variants" *Machine Learning* 36(1-2) pp 105-139.
- Bernado-Mansilla E, and Garrell-Guiu JM, 2003. "Accuracy-based learning classifier systems: models, analysis and applications to classification tasks" *Evolutionary Computation* 11(3) pp 209-238.
- Bersano-Begey TF, and Daida JM, 1997. "A discussion on generality and robustness and a framework for fitness set construction in genetic programming to promote robustness" *Genetic Programming 1997: Late Breaking Papers of the 2nd Annual Conference* pp 11-18.
- Bezdek JC, 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press.
- Bonabeau E, Dorigo M, and Theraulaz G, 1999. *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press.
- Bonarini A, 2000. "An introduction to learning fuzzy classifier systems" *Lecture Notes in Artificial Intelligence 1813*, Lanzi PL *et al.* (Ed.), Berlin Heidelberg: Springer-Verlag.
- Bonelli P, and Parodi A, 1991. "An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains" *Proceedings of the 4th International Conference on Genetic Algorithms* pp 288-295.
- Breiman L, 1996. "Bagging predictors" *Machine Learning* 24(2) pp 123-140.
- Bucci A, and Pollack JB, 2002. "A mathematical framework for the study of coevolution" *Foundations of Genetic Algorithms VII*.
- Buchanan BG, and Duda R, 1983. "Principles of rule-based expert systems" *Advances in Computers* 22 pp 163-216.
- Cantu-Paz E, and Kamath C, 2003. "Inducing oblique decision trees with evolutionary algorithms" *IEEE Transactions on Evolutionary Computation* 7(1) pp 54-68.
- Carse B, and Pipe AG, 2002. "On generalisation in Michigan-style fuzzy classifier systems" Report: UWLCS02-006, UWE Learning Classifier Systems Group, Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England.
- Casillas J, Cordon O, Herrera F, and Merelo JJ, 2002. "Cooperative coevolution for learning fuzzy rule-based systems" *Lecture Notes in Computer Science 2310*, Springer-Verlag.

- Cavaretta MJ, and Chellapilla K, 1999. "Data mining using genetic programming: the implications of parsimony on generalization error" *Proceedings of the 1999 Congress on Evolutionary Computation* pp 1330-1337.
- Chen SM, Ke JS, and Chang JF, 1990. "Knowledge representation using fuzzy petri nets" *IEEE Transactions on Knowledge and Data Engineering* 2 pp 311-319.
- Clark P, and Niblett T, 1989. "The CN2 induction algorithm" *Machine Learning* 3(4) pp 261-283.
- Coello Coello CA, 1999. "A comprehensive survey of evolutionary-based multiobjective optimization techniques" *Knowledge and Information Systems* 1(3) pp 129-156.
- Cordon O, del Jesus MJ, and Herrera F, 1998. "Genetic learning of fuzzy rule-based classification systems co-operating with fuzzy reasoning methods" *International Journal of Intelligent Systems* 13 pp 1025-1053.
- Cordon O, del Jesus MJ, Herrera F, and Lozano M, 1999. "MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach" *International Journal of Intelligent Systems* 14(11) pp 1123-1153.
- Cordon O, Gonzalez A, Herrera F, and Perez R, 1998. "Encouraging cooperation in the genetic iterative rule learning approach for qualitative modeling" *Computing with Words in Intelligent/Information Systems*, Kacprzyk J and Zadeh L (Eds.), Physica-Verlag.
- Cordon O, Herrera F, Gomide F, Hoffmann F, and Magdalena L (Guest Eds.), 2004. "Genetic Fuzzy Systems: New Developments" special issue of *Fuzzy Sets and Systems* 141(1).
- Davis R, and King J, 1977. "An overview of production systems" *Proceedings of the 8th Machine Intelligence Workshop* pp 300-332.
- De Jong KA, Spears W, M., and Gordon DF, 1993. "Using genetic algorithms for concept learning" *Machine Learning* 13 pp 161-188.
- Dietterich TG, 2000. "Ensemble methods in machine learning" *Lecture Notes in Computer Science 1857*, Kittler J and Roli F (Eds.), Springer-Verlag.
- Divina F, and Marchiori E, 2002. "Evolutionary concept learning" *Proceedings of the Genetic and Evolutionary Computation Conference* pp 343-350.
- Donnart J, and Meyer J, 1994. "A hierarchical classifier system implementing a motivationally autonomous animat" *Proceedings of the 3rd International Conference on Simulation of Adaptive Behaviour* pp 144-153.
- Dorigo M, 1995. "AlecSys and the autoNOMouse: learning to control a real robot by distributed classifier systems" *Machine Learning* 19 pp 209-240.
- Drobits M, Bodenhofer U, and Klement EP, 2003. "FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions" *International Journal of Approximate Reasoning* 32(2-3) pp 131-152.
- Ebrahim R, 2001. "Fuzzy logic programming" *Fuzzy Sets and Systems* 117(2) pp 215-230.
- Eggermont J, 2002. "Evolving fuzzy decision trees with genetic programming and clustering" *Lecture Notes in Computer Science 2278*, Foster JA et al. (Ed.), Berlin Heidelberg: Springer-Verlag.
- Endou T, and Zhao Q, 2002. "Generation of comprehensible decision trees through evolution of training data" *Proceedings of the 2002 IEEE World Congress on Computational Intelligence* pp 1221-1225.
- Fogel DB, 1997. "The advantages of evolutionary computation" *Biocomputing and Emergent Computation 1997* pp 1-11.

- Fogel LJ, Owens AJ, and Walsh MJ, 1966. *Artificial Intelligence Through Simulated Evolution*, New York: John Wiley & Sons.
- Freitas AA, 2002. "A survey of evolutionary algorithms for data mining and knowledge discovery" *Advances in Evolutionary Computation*, Ghosh A and Tsutsui S (Eds.), Springer-Verlag.
- Freitas AA, 1999. "A genetic algorithm for generalized rule induction" *Advances in Soft Computing - Engineering Design and Manufacturing*, R.Roy *et al.* (Ed.), Springer-Verlag.
- Freitas AA, 1998. "On objective measures of rule surprisingness" *Lecture Notes in Artificial Intelligence 1510*, Berlin Heidelberg: Springer-Verlag.
- Freund Y, and Schapire RE, 1996. "Experiments with a new boosting algorithm" *Proceedings of the 13th International Conference on Machine Learning* pp 148-156.
- Garrell i Guiu J, Golobardes i Ribe E, Bernado i Mansilla E, and Llorca i Fabrega F, 1998. "Automatic classification of mammary biopsy images with machine learning techniques" *Proceedings of Engineering of Intelligent Systems* pp 411-418.
- Gath I, and Geva B, 1989. "Unsupervised optimal fuzzy clustering" *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11(7) pp 773-781.
- Gent IP, Grant SA, MacIntyre E, Prosser P, Shaw P, Smith BM, and Walsh T, 1997. "How not to do it" Report: 97.27, School of Computer Studies, University of Leeds, Leeds, England.
- Giordana A, and Saitta L, 1993. "REGAL: an integrated system for learning relations using genetic algorithms" *Proceedings of the 2nd International Workshop on Multistrategy Learning* pp 234-249.
- Gomez-Skarmeta AF, Jimenez F, and Ibaez J, 1998. "Pareto-optimality in fuzzy modeling" *6th European Congress on Intelligent Techniques and Soft Computing* pp 694-700.
- Gomez-Skarmeta AF, Valdes M, Jimenez F, and Marin-Blazquez JG, 2001. "Approximative fuzzy rules approaches for classification with hybrid-GA techniques" *Information Sciences* 136(1-4) pp 193-214.
- Gonzalez A, 1995. "A learning methodology in uncertain and imprecise environments" *International Journal of Intelligent Systems* 19 pp 357-371.
- Gonzalez A, and Herrera F, 1997. "Multi-stage genetic fuzzy systems based on the iterative rule learning approach" *Mathware and Soft Computing* 4 pp 233-249.
- Gonzalez A, and Perez R, 1999. "SLAVE: A genetic learning system based on an iterative approach" *IEEE Transactions on Fuzzy Systems* 7(2) pp 176-191.
- Gonzalez A, and Perez R, 1998. "Completeness and consistency conditions for learning fuzzy rules" *Fuzzy Sets and Systems* 96 pp 37-51.
- Gonzalez A, Perez R, and Verdegay JL, 1994. "Learning the structure of a fuzzy rule: A genetic approach" *Fuzzy Systems and Artificial Intelligence* 3(1) pp 57-70.
- Guerra-Salcedo C, and Whitley D, 1999. "Genetic approach for feature selection for ensemble creation" *Proceedings of the Genetic and Evolutionary Computation Conference* pp 236-243.
- Hansen L, and Salamon P, 1990. "Neural network ensembles" *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 pp 993-1001.
- Harris R, 2002. "Fuzzifying GABIL: evolving a fuzzy rulebase for adaptive machine learning", Graduate research paper, University of Nevada, Reno.
- Hekanaho J, 1998. "DOGMA: a GA based relational learner" *Proceedings of the 8th International Conference on Inductive Logic Programming* pp 205-214.
- Hilderman RJ, and Hamilton HJ, 1999. "Heuristics for ranking the interestingness of discovered



- knowledge" *Lecture Notes in Artificial Intelligence 1574*, Berlin Heidelberg: Springer-Verlag.
- Hirota K (Ed.), 1993. *Industrial Applications of Fuzzy Technology*, Springer-Verlag.
- Hoffmann F, 2004. "Combining boosting and evolutionary algorithms for learning of fuzzy classification rules" *Fuzzy Sets and Systems* 141(1) pp 47-58.
- Hoffmann F, 2001. "Evolutionary algorithms for fuzzy control system design" *Proceedings of the IEEE* 89(9) pp 1318-1333.
- Holland JH, 1975. *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press.
- Holland JH, Booker LB, Colombetti M, Dorigo M, Goldberg DE, Forrest S, Riolo RL, Smith RE, Lanzi PL, Stolzmann W, and Wilson SW, 2000. "What is a learning classifier system?" *Lecture Notes in Computer Science 1813*, Springer Verlag.
- Holland JH, and Reitman JS, 1978. "Cognitive systems based on adaptive algorithms" *Pattern-directed Inference Systems*, Waterman DA and Hayes-Roth F (Eds.), NY: Academic Press.
- Holmes JH, 2000. "Learning classifier systems applied to knowledge discovery in clinical research databases" *Lecture Notes in Artificial Intelligence 1813*, Lanzi PL *et al.* (Ed.), Springer-Verlag.
- Hsu WW, and Hsu C, 2002. "GEC: an evolutionary approach for evolving classifiers" *Lecture Notes in Artificial Intelligence 2336*, Chen M- *et al.* (Ed.), Berlin Heidelberg: Springer-Verlag.
- Ishibuchi H, Nakashima T, and Kuroda T, 2000. "A hybrid fuzzy GBML algorithm for designing compact fuzzy rule-based classification systems" *Proceedings of the 9th IEEE International Conference on Fuzzy Systems* pp 706-711.
- Ishibuchi H, Nakashima T, and Kuroda T, 1999a. "A fuzzy genetics-based machine learning method for designing linguistic classification systems with high comprehensibility" *Proceedings of the 6th International Conference on Neural Information Processing* pp 597-602.
- Ishibuchi H, Nakashima T, and Murata T, 1999b. "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems" *IEEE Transactions on Systems, Man and Cybernetics, Part B* 29(5) pp 601-618.
- Ishibuchi H, Nakashima T, and Murata T, 1996. "Genetic-algorithm-based approaches to the design of fuzzy systems for multi-dimensional pattern classification problems" *Proceedings of 1996 IEEE International Conference on Evolutionary Computation* pp 229-234.
- Ishibuchi H, Nakashima T, and Murata T, 1995. "A fuzzy classifier system that generates fuzzy if-then rules for pattern classification problems" *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation* pp 759-764.
- Ishibuchi H, and Yamamoto T, 2004. "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining" *Fuzzy Sets and Systems* 141(1) pp 59-88.
- Janikow CZ, 1996. "A genetic algorithm method for optimizing fuzzy decision trees" *Information Sciences* 89 pp 275-296.
- Janikow CZ, 1993. "A knowledge-intensive genetic algorithm for supervised learning" *Machine Learning* 13(2-3) pp 189-228.
- Jeong J, and Oh S, 1999. "Automatic rule generation for fuzzy logic controllers using rule-level co-evolution of subpopulations" *Proceedings of the 1999 Congress on Evolutionary Computation (CEC-1999)* pp 2151-2156.
- Jimenez F, Gomez-Skarmeta AF, Roubos H, and Robert B, 2001. "Accurate, transparent, and compact fuzzy models for function approximation and dynamic modelling through multi-objective evolutionary optimization" *1st International Conference on Evolutionary Multi-*

- Criterion Optimization* pp 653-667.
- Juille H, and Pollack JB, 1998. "Coevolutionary learning: a case study" *Proceedings of the 15th International Conference on Machine Learning*.
- Junco L, and Sanchez L, 2000. "Using the adaboost algorithm to induce fuzzy rules in classification problems" *Proceedings of the Spanish Conference for Fuzzy Logic and Technologies* pp 297-301.
- Kainz G, and Kaindl H, 1998. "Towards a standardized comparison of search algorithms" available from: [citeseer.nj.nec.com/12011.html](http://citeseer.nj.nec.com/12011.html).
- Keijzer M, and Babovic V, 2000. "Genetic programming, ensemble methods and the bias/variance tradeoff - introductory investigations" *Lecture Notes in Computer Science 1802*, Berlin Heidelberg: Springer-Verlag.
- Kim Y, Street WN, and Menczer F, 2002. "Meta-evolutionary ensembles" *Proceedings of the IEEE International Joint Conference on Neural Networks* pp 2791-2796.
- Kovacs T, 2001. "Two views of classifier systems" *Fourth International Workshop on Learning Classifier Systems* pp 367-371.
- Koza JR, 1992. *Genetic Programming*, Bradford Book, The MIT Press.
- Koza JR, 1991. "Concept formation and decision tree induction using the genetic programming paradigm" *Parallel Problem Solving from Nature*, Schwefel HP and Manner R (Eds.), Berlin, Germany: Springer-Verlag.
- Krishnapuram R, and Keller JM, 1993. "A possibilistic approach to clustering" *IEEE Transactions on Fuzzy Systems* 1(2) pp 98-110.
- Kushchu I, 2002. "An evaluation of evolutionary generalisation in genetic programming" *Artificial Intelligence Review* 18(1) pp 3-14.
- Kushchu I, 2000. "Generalisation and domain specific functions in genetic programming" *Proceedings of the 2000 Congress on Evolutionary Computation* pp 1393-1400.
- Kwedlo W, and Kretowski M, 1998. "Discovery of decision rules from databases: an evolutionary approach" *Lecture Notes in Artificial Intelligence 1510*, Berlin: Springer-Verlag.
- Langdon WB, and Buxton BF, 2001. "Genetic programming for combining classifiers" *Proceedings of the Genetic and Evolutionary Computation Conference*.
- Langley P, 2000. "Crafting papers on machine learning" *Proceedings of the 17th International Conference on Machine Learning* pp 1207-1211.
- Lanzi PL, and Riolo RL, 2000. "A roadmap to the last decade of learning classifier system research (from 1989 to 1999)" *Lecture Notes in Artificial Intelligence 1813*, Lanzi PL et al. (Eds.), Berlin Heidelberg: Springer-Verlag.
- Lavrac N, 1998. "Computational logic and machine learning: A roadmap for inductive logic programming" Technical Report, J. Stefan Institute, Ljubljana, Slovenia.
- Leung K, King I, and Tse M, 1999. "FF99: a novel fuzzy first-order logic learning system" *Proceedings of the IEEE Conference on Systems, Man and Cybernetics* pp 178-184.
- Li D, and Liu D, 1990. *A Fuzzy Prolog Database System*, John Wiley & Sons.
- Ling CX, Huang J, and Zhang H, 2003. "AUC: A better measure than accuracy in comparing learning algorithms" *Lecture Notes in Artificial Intelligence 2671*, Xiang Y and Chaib-draa B (Eds.), Berlin Heidelberg: Springer-Verlag.
- Liu B, Hsu W, and Chen S, 1997. "Using general impressions to analyze discovered classification rules" *Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining*

(KDD-97) pp 31-36.

- Liu JJ, and Kwok JT, 2000. "An extended genetic rule induction algorithm" *Proceedings of the 2000 Congress on Evolutionary Computation* pp 458-463.
- Llora X, and Garrell JM, 2001. "Evolution of decision trees" *Proceedings of the 4th Catalan Conference on Artificial Intelligence*.
- Llora X, Goldberg DE, Traus I, and Bernado E, 2002. "Accuracy, parsimony, and generality in evolutionary learning systems via multiobjective selection" Report: IlliGAL 2002016, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.
- Mamdani EH, 1976. "Advances in the linguistic synthesis of fuzzy controllers" *Journal of Man-Machine Studies* 8 pp 669-678.
- Marin-Blazquez JG, and Shen Q, 2002. "From approximate to descriptive fuzzy classifiers" *IEEE Transactions on Fuzzy Systems* 10(4) pp 484-497.
- Martienne E, and Quafafou M, 1998. "Learning fuzzy relational descriptions using the logical framework and rough set theory" *Proceedings of the 7th IEEE International Conference on Fuzzy Systems* pp 944.
- Mayer HA, Furlinger K, and Strapetz M, 1999. "Extraction of compact rule sets from evolutionary designed artificial neural networks" *IEEE Systems, Man and Cybernetics Conference* pp 424.
- Mendes R, Voznika FdB, Freitas AA, and Nievola JC, 2001. "Discovering fuzzy classification rules with genetic programming and co-evolution" *Lecture Notes in Artificial Intelligence 2168*, Springer-Verlag.
- Michalski RS, 1969. "On the quasi-minimal solution of the covering problem" *Proceedings of the 5th International Symposium on Information Processing* pp 125-128.
- Michie D, Spiegelhalter DJ and Taylor CC (Eds.), 1994. *Machine Learning, Neural and Statistical Classification*, New York: Ellis Horwood.
- Muggleton S, 1995. "Inverse entailment and prolog" *New Generation Computing* 13(3-4) pp 245-286.
- Muggleton S, and De Raedt L, 1994. "Inductive logic programming" *The Journal of Logic Programming* 19/20 pp 629-679.
- Muggleton S, and Feng C, 1990. "Efficient induction of logic programs" *Proceedings of the 1st Conference on Algorithmic Learning Theory*.
- Munakata T, 1998. "Notes on implementing fuzzy sets in prolog" *Fuzzy Sets and Systems* 98(3) pp 311-317.
- Murthy SK, Kasif S, and Salzberg S, 1994. "A system for induction of oblique decision trees" *Journal of Artificial Intelligence Resources* 2(1) pp 1-32.
- Nadeau C, and Bengio Y, 2003. "Inference for the generalization error" *Machine Learning* 52 pp 239-281.
- Nakashima T, Ishibuchi H, and Murata T, 1998. "Evolutionary algorithms for constructing linguistic rule-based systems for high-dimensional pattern classification problems" *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation* pp 752-757.
- Noda E, Freitas AA, and Lopes HS, 1999. "Discovering interesting prediction rules with a genetic algorithm" *Proceedings of the Congress on Evolutionary Computation* pp 1322-1329.
- Noda E, Freitas AA, and Yamakami A, 2002. "A distributed-population genetic algorithm for discovering interesting prediction rules" *Proceedings of the 7th Online World Conference on Soft Computing in Industrial Applications* pp 368-381.

- Pareto V, 1896. *Cours d'Economie Politique*, Lausanne: F. Rouge.
- Parmanto B, Munro PW, and Doyle HR, 1996. "Improving committee diagnosis with resampling techniques" *Advances in Neural Information Processing Systems* 8 pp 882-888.
- Parsons S, 2001. *Qualitative Methods for Reasoning under Uncertainty*, Cambridge, Massachusetts: The MIT Press.
- Pedrycz W (Ed.), 1996. *Fuzzy Modelling: Paradigms and Practice*, Kluwer Academic Press.
- Pena-Reyes CA, and Sipper M, 2001. "FuzzyCoCo: A cooperative-coevolutionary approach to fuzzy modeling" *IEEE Transactions on Fuzzy Systems* 9(5) pp 727-737.
- Pena-Reyes CA, and Sipper M, 1999. "A fuzzy-genetic approach to breast cancer diagnosis" *Artificial Intelligence in Medicine* 17(2) pp 131-155.
- Potter MA, and De Jong KA, 2000. "Cooperative coevolution: an architecture for evolving coadapted subcomponents" *Evolutionary Computation* 8(1) pp 1-29.
- Quinlan JR, 1992. *C4.5: Programs for Machine Learning*, CA: Morgan Kaufmann.
- Quinlan JR, 1990. "Learning logical definitions from relations" *Machine Learning* 5(3) pp 239-266.
- Quinlan JR, 1986. "Induction on decision trees" *Machine Learning* 1 pp 81-106.
- Ragot N, and Anquetil E, 2001. "A new hybrid learning method for fuzzy decision trees" *Proceedings of the 10th IEEE International Conference on Fuzzy Systems* pp 1380-1383.
- Rechenberg I, 1973. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Stuttgart: Fromman-Holzboog Verlag.
- Riquelme JC, Aguilar JS, and Toro M, 2000. "Discovering hierarchical decision rules with evolutive algorithms in supervised learning" *International Journal of Computers, Systems and Signals* 1(1) pp 73-84.
- Romao W, Freitas AA, and Pacheco RCS, 2002. "A genetic algorithm for discovering interesting fuzzy prediction rules: applications to science and technology data" *Proceedings of the Genetic and Evolutionary Computation Conference* pp 343-350.
- Rosin CD, and Belew RK, 1997. "New methods for competitive coevolution" *Evolutionary Computation* 5(1) pp 1-29.
- Rouwhorst SE, and Engelbrecht AP, 2000. "Searching the forest: using decision tree as building blocks for evolutionary search in classification" *Proceedings of the 2000 Congress on Evolutionary Computation* pp 633-638.
- Rowland JJ, 2003. "Generalisation and model selection in supervised learning with evolutionary computation" *Lecture Notes in Computer Science 2611*, Berlin Heidelberg: Springer Verlag.
- Setiono R, 2000. "Generating concise and accurate classification rules for breast cancer diagnosis" *Artificial Intelligence in Medicine* 18(3) pp 205-219.
- Shibata D, Inuzuka N, Kato S, Matsui T, and Itoh H, 1999. "An induction algorithm based on fuzzy logic programming" *Lecture Notes in Artificial Intelligence 1574*, Zhong N and Zhou L (Eds.), Berlin Heidelberg: Springer-Verlag.
- Silberschatz A, and Tuzhilin A, 1996. "What makes patterns interesting in knowledge discovery systems" *IEEE Transactions on Knowledge and Data Engineering* 8(6) pp 970-974.
- Sirlantzis K, Fairhurst MC, and Guest RM, 2002. "An evolutionary algorithm for classifier and combination rule selection in multiple classifier systems" *Proceedings of the International Conference on Pattern Recognition* pp 771-774.
- Slawinski T, Krone A, Hammel U, Wiesmann D, and Krause P, 1999. "A hybrid evolutionary search

- concept for data-based generation of relevant fuzzy rules in high dimensional spaces" *Proceedings of the IEEE International Conference on Fuzzy Systems* pp 1432-1437.
- Smith MG, and Bull L, 2003. "Feature construction and selection using genetic programming and a genetic algorithm" *Lecture Notes in Computer Science*, Ryan C *et al.* (Ed.),.
- Smith SF, 1980. *A Learning System Based on Genetic Adaptive Algorithms*, PhD Dissertation, Computer Science Department, University of Pittsburgh.
- Stolzmann W, 2001. "Anticipatory classifier systems: an introduction" *AIP Conference Proceedings* 573 pp 470-476.
- Suzuki E, and Kodratoff Y, "Discovery of surprising exception rules based on intensity of implication" *Lecture Notes in Artificial Intelligence 1510*, Berlin Heidelberg: Springer Verlag.
- Takagi T, and Sugeno M, 1985. "Fuzzy identification of systems and its application to modeling and control" *IEEE Transactions on Systems, Man and Cybernetics* 15 pp 116-132.
- Thompson S, 1999. "Genetic algorithms as postprocessors for data mining" *Data Mining with Evolutionary Algorithms: Research Directions - Papers from the AAAI Workshop*.
- Tufts P, 1995. "Dynamic classifiers: genetic programming and classifier systems" *Working Notes for the AAAI Symposium on Genetic Programming* pp 114-119.
- Umanol M, Okamoto H, Hatono I, Tamura H, Kawachi F, Umedzu S, and Kinoshita J, 1994. "Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems" *Proceedings of the IEEE Conference on Fuzzy Systems* pp 2113-2118.
- Van Veldhuizen DA, and Lamont GB, 2000. "Multiobjective evolutionary algorithms: analyzing the state-of-the-art" *Evolutionary Computation* 8(2) pp 125-147.
- Venturini G, 1993. "SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts" *Proceedings of the European Conference on Machine Learning* pp 296.
- Walter D, and Mohan CK, 2000. "ClaDia: A fuzzy classifier system for disease diagnosis" *Proceedings of the Congress on Evolutionary Computation* pp 1429-1435.
- Watson R, and Pollack J, 2001. "Coevolutionary dynamics in a minimal substrate" *Proceedings of the Genetic and Evolutionary Computation Conference* pp 702-709.
- Whitley D, 2001. "An overview of evolutionary algorithms: practical issues and common pitfalls" *Information and Software Technology* 43(14) pp 817-831.
- Wolfram S, 1983. "Cellular automata" *Los Alamos Science* 9(Fall 1983) pp 2-21.
- Wong ML, 2001. "A flexible knowledge discovery system using genetic programming and logic grammars" *Decision Support Systems* 31 pp 405-428.
- Wong ML, and Leung KS, 1995a. "Inducing logic programs with genetic algorithms - the genetic logic programming system" *IEEE Expert-Intelligent Systems and their Applications* 10(5) pp 68-76.
- Wong ML, and Leung KS, 1995b. "An induction system that learns programs in different programming languages using genetic programming and logic grammars" *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*.
- Yang L, Widyantoro DH, Ioerger T, and Yen J, 2001. "An entropy-based adaptive genetic algorithm for learning classification rules" *Proceedings of the Congress on Evolutionary Computation* pp 790-796.
- Yao X, Liu Y, and Darwen P, 1996. "How to make best use of evolutionary learning" *Complex Systems - From Local Interactions to Global Phenomena* pp 229-242.

- Yao X, 1999. "Evolving artificial neural networks" *Proceedings of the IEEE* 87(9) pp 1423-1447.
- Yuan YF, and Zhuang H, 1996. "A genetic algorithm for generating fuzzy classification rules" *Fuzzy Sets and Systems* 84(1) pp 1-19.
- Yuhui S, Eberhart R, and Yaobin C, 1999. "Implementation of evolutionary fuzzy systems" *IEEE Transactions on Fuzzy Systems* 7(2) pp 109-119.
- Zadeh LA, 1988. "Fuzzy logic" *IEEE Computer* 21(4) pp 83-92.
- Zadeh LA, 1965. "Fuzzy sets" *Information Control* 8 pp 338-353.
- Zorman M, Podgorelec V, Kokol P, Peterson M, and Lane J, 2000. "Decision tree's induction strategies evaluated on a hard real world problem" *Proceedings of the 13th IEEE Symposium on Computer-Based Medical Systems* pp 19-24.