Fuzzy rule weight modification with particle swarm optimisation

Chen, Tianhua; Shen, Qiang; Su, Pan; Shang, Changjing

Published in:
Soft Computing

DOI:
10.1007/s00500-015-1922-z

Publication date:
2016

Citation for published version (APA):
Chen, T., Shen, Q., Su, P., & Shang, C. (2016). Fuzzy rule weight modification with particle swarm optimisation. *Soft Computing*, *20*(8), 2923–2937. https://doi.org/10.1007/s00500-015-1922-z

Noname manuscript No.
(will be inserted by the editor)

Fuzzy Rule Weight Modification with Particle Swarm Optimisation

Tianhua Chen · Qiang Shen · Pan Su · Changjing Shang

Received: date / Accepted: date

Abstract The most challenging problem in developing fuzzy rule-based classification systems is the construction of a fuzzy rule base for the target problem. In many practical applications, fuzzy sets that are of particular linguistic meanings, are often predefined by domain experts and required to be maintained in order to ensure interpretability of any subsequent inference results. However, learning fuzzy rules using fixed fuzzy quantity space without any qualification will restrict the accuracy of the resulting rules. Fortunately, adjusting the weights of fuzzy rules can help improve classification accuracy without degrading the interpretability. There have been different proposals for fuzzy rule weight tuning through the use of various heuristics with limited success. This paper proposes an alternative approach using Particle Swarm Optimisation in the search of a set of optimal rule weights, entailing high classification accuracy. Systematic experimental studies are carried out using common benchmark data sets, in comparison to popular rule based learning classifiers. The results demonstrate that the proposed approach can boost classification performance, especially when the size of initially built rule base is relatively small, and is competitive to popular rule based learning classifiers.

Tianhua Chen · Qiang Shen · Pan Su · Changjing Shang
Department of Computer Science, Institute of Mathematics, Physics and Computer Science, Aberystwyth University, UK
E-mail: cns@aber.ac.uk

T. Chen
E-mail: tic4@aber.ac.uk

Q. Shen
E-mail: qqs@aber.ac.uk

P. Su
E-mail: pas23@aber.ac.uk

Keywords Fuzzy rule induction · Fuzzy rule weights · Rule weight modification · Particle swarm optimisation

1 Introduction

Fuzzy rule induction forms a major approach to learning robust transparent classification models. The use of such learning algorithms allows for enhanced transparency in both the learned models themselves and the inferences performed with the resulting learning classifiers. A fuzzy rule-based classification system (FRBCS) is a special case of fuzzy modelling where the output of the system is crisp and discrete. Many approaches have been proposed for generating and learning fuzzy if-then rules from numerical data to model the input-output behaviour of a certain problem. These include a variety of Genetic Algorithm based methods, with Michigan-style representation [1,2] that denotes a single fuzzy rule as an individual chromosome, and Pittsburgh-style representation [3,4] that expresses the entire rule base as an individual. Other popular fuzzy rule induction methods include fuzzy association rule mining [5], neuro-fuzzy techniques [6], rough-fuzzy or fuzzy-rough based methods [7–9], and linguistic semantics-preserving modelling [10, 11].

The techniques developed in the literature have had success in their carefully selected applications. However, a major challenge in learning FRBCSs remains for situations where the membership functions defining the antecedent fuzzy sets are prefixed, with each having a specific linguistic meaning pre-specified by domain experts (and typically also known to the user). Due to the need of maintaining the interpretability of a learned model, any learned fuzzy classification rule is required to use these fuzzy sets to specify the values of the at-

tributes. Yet, using a fixed quantity space consisting of such given fuzzy sets limits the accuracy of the learnt rules. Fortunately, this problem can be tackled by modifying the weights associated with the individual rules.

Rule weights intuitively reveal the relative importance amongst all the rules in a given rule base. The greater the rule weight of a fuzzy if-then rule, the more likely it will be chosen to classify an unseen pattern amongst all the fuzzy rules that cover the subspace of that pattern. The modification of a rule's weight is in effect equivalent to the adjustment of the membership functions of those antecedent fuzzy sets in the rule [12]. Interestingly, the adjustment of rule weights is much easier than directly modifying the antecedent fuzzy sets (which would involve the learning of a number of parameters for each membership function), since there is only one single parameter (namely, the weight itself) per rule to learn [13]. It is therefore desirable to improve the performance of a given rule base by carefully adjusting the rule weights, instead of learning a set of dynamic membership functions.

In [14], a seminal method of leaning rule weights is proposed by the use of an error correction-based learning procedure with post-learning pruning, through a "Reward and Punishment" scheme. It works by increasing the weight when a pattern is correctly classified by the current rule, and decreasing the rule weight otherwise. Another weighting approach is reported in [15], by dividing the covering subspace of each fuzzy rule into two subdivisions based on a given threshold. The association degree of any pattern with a so-called compatibility grade above the threshold is enhanced by increasing the weight. The splitting threshold for each rule is found by exploiting the distribution of patterns in the subspace covered by that rule. Other rule weight learning methods for building FRBCSs include [13] and [16]. The importance and effects of learning rule weights in FRBCSs have been discussed and highlighted in [17], and a number of heuristic methods for fuzzy rule weight specifications can also be found in [18].

The performance of a particular fuzzy rule may be improved by directly adjusting its rule weight. However, the performance of its neighbouring fuzzy rules (i.e., those that also cover the same given pattern) may be deteriorated or even become useless due to the propagation of such modifications to the rest of the rule base. The overall consequence is thus unpredictable when all the rule weights are changing successively. Instead of solely using heuristic weighting functions to tune fuzzy if-then rule weights, based on the preliminary investigation that was reported in [19], this paper describes the development of an evolutionary algorithm-based approach to modifying rule weights in FRBCSs. This is inspired by the observeation that evolutionary algorithms often perform well in globally approximating optimal solutions to various types of problem [20–24]. In particular, Particle Swarm Optimisation (PSO) [25] is employed as the evolutionary algorithm to evolve rule weights in order to improve the classification accuracy.

In this work, to take a data-driven approach, the generation of an initial rule base is done by straightforward fuzzy grid partitioning for each input dimension. Each partition of the input space is identified by a fuzzy rule if there is at least one training pattern in that subspace [26]. However, in general, if domain expertise is available, expert-provided rules may be used as the initial rule base. The proposed method encodes fuzzy rule weights as particle dimensions with the initial population consisting of heuristically created rule weights [27], covering each class of the given problem and using predefined partitions for all linguistic variables. Classification accuracy is set as the fitness evaluation required. The strategy of "single winner rule" [28] (i.e., the rule that leads to the highest classification accuracy) is adopted to choose which rule to perform classification, due to its simplicity and intuitive appeal. A dozen of benchmark UCI data sets are employed to examine the performance of the proposed approach, in comparison with popular rule based learning classifier algorithms, including C4.5 decision tree [29], top-down fuzzy pattern trees [30], and fuzzy subsethood-based rule models with quantifiers [31]. Analysis of the proposed approach in terms of overfitting is made with respect to both the size of the initial rule base and the convergence time of the learning process.

The reminder of this paper is organised as follows. Section II describes the heuristic methods adopted for the generation of an initial fuzzy rule set and for the specification of rules weights. For completeness it also introduces the reasoning method used for performing classification of new patterns, together with an outline of particle swarm optimisation. Section III demonstrates how the rule weights may affect the classification boundaries, and how PSO is employed to evolve rule weights in order to obtain an optimal solution, including a complexity analysis of the proposed approach. Section IV presents and discusses the experimental results. Section V concludes the paper and points out ideas for further development.

2 Preliminaries

This section first describes the heuristic methods adopted for the generation of an initial fuzzy rule set as well as the specification of initially generated rules weights.

The reasoning method employed for classification by firing learnt fuzzy rules is also briefly explained, followed by a brief introduction to particle swarm optimisation.

2.1 Fuzzy Rule Learning

The task of learning from or generalising a given problem description, by the use of fuzzy logic and fuzzy sets, is to find a finite set of fuzzy if-then rules capable of reproducing the input-output behaviour of a given system (or process). Without losing generality, the system to be learnt is herein assumed to be a multiple-input-single-output, containing n inputs and one output and involving m patterns for an M-class problem. A fuzzy if-then rule $R_j, j = 1, 2, ..., N$, for such a system is represented as follows:

$$\text{If } x_1 \text{ is } A_{j1} \text{ and } ... \text{ and } x_n \text{ is } A_{jn} \text{ then class is } C_h \text{ with } w_j \tag{1}$$

where $x_1, x_2, ..., x_n$ are the underlying linguistic variables, jointly defining an n-dimensional pattern space; A_{ji}, $i \in \{1, 2, ..., n\}$, is the fuzzy value of the corresponding antecedent x_i; C_h, $h \in \{1, 2, ..., M\}$, is the consequent class for the M class problem; and w_j is the rule weight of fuzzy rule R_j indicating the strength that any input pattern $X_p = [x_{p1}, x_{p2}, ..., x_{pn}]$, $p \in \{1, 2, ..., m\}$ within the fuzzy subspace delimited by the given antecedent values is deemed to belong to the consequent class C_h.

In order to obtain an initial set of fuzzy if-then rules, the domain of each attribute is partitioned into K ($K \geq 2$) subsets $\{A_1^K, A_2^K, ..., A_K^K\}$. Practically speaking, partitioning the input space and defining the corresponding fuzzy sets are typically done by the domain experts (even though such specification may reflect a certain biased view of particular individuals). Of course, the performance of a resulting learnt classifier may vary in relation to the variation of the partition of the input space, especially regarding the number of the partitions made. When the fuzzy partition is too coarse in the sense that the number of generated fuzzy subspaces is too small, the testing data may not be covered by the resulting rules. On the other hand, if the partitioning of the fuzzy subspace is too fine such that the number of generated fuzzy subspaces is too large, there may not be sufficient data points to support the training [26]. Moreover, the finer the partition is, the more likely that more rules will be generated in the initial rule base, which will in turn lead to more complex computation in achieving the classification task using the resultant rules. The impact of the size of a rule base upon the performance of a learning classifier will be further investigated later by examining the effects of using different partitions of a given problem domain.

In this work, for simplicity and also for having an unbiased, common footing to perform subsequent comparative studies, the following assumption is made: The two domain-delimiting values of each dimension are defined as rectangular triangular fuzzy sets, and the rest of the dimension is divided equally into $(K - 2)$ fuzzy regions, with the corresponding fuzzy membership values represented by the symmetric triangular functions as shown in Fig. 1. Note however, that this assumption is not necessary in applying the underlying techniques proposed here as any given initial fuzzy petition may be used to form the initial rule base. In Fig. 1, a and b represent the minimum and maximum value of x_{pi} taken from the training examples, respectively. The vertex location of a symmetric triangular is calculated according to its position within the $(K - 2)$ partitions. Membership values of x_{pi} in a new pattern below a or above b are set to 1. Following the general principle of data-driven learning, each partition is identified by a fuzzy rule if there is at least one training pattern in that pattern subspace [26]. That is, given an input partitioning of pattern space, a fuzzy rule will be generated only if there is a training pattern covered by this rule. Thus, for a problem with m training patterns, at most m rules may be generated.

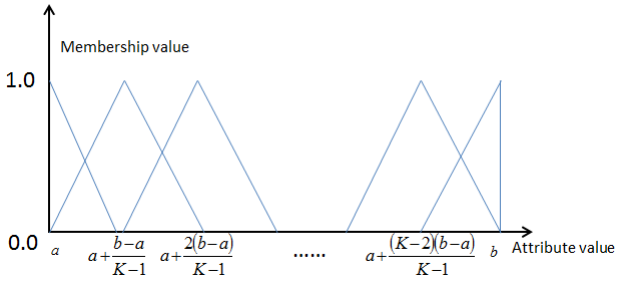

Fig. 1 Partitioning of each pattern space dimension

There are a number of different approaches to specifying fuzzy rule weights [18]. This work adopts the classical method of [27] owing to its maturity. Following this approach, the consequent class C_h of fuzzy rule R_j and the corresponding rule weight w_j are determined by the following procedure, where rule generation is a direct by-product:

1. Calculate the matching degree for each class C_h with respect to the possible antecedents, which is defined by

$$\beta_{C_h} = \sum_{X_p \in C_h} \prod_{i=1}^{n} \mu_{A_{ji}}(x_{pi}) \tag{2}$$

where X_p are the training patterns defined on the corresponding n-dimensional fuzzy subspace $A_j = A_{j1} \times A_{j2} \times \cdots \times A_{jn}$, and $\mu_{A_{ji}}(\cdot)$ is the membership function of the antecedent fuzzy set A_{ji}.

2. Find $\beta_{C_T}, T = 1, 2, ..., M$, such that

$$\beta_{C_T} = max\{\beta_{C_1}, \beta_{C_2}, ...\beta_{C_M}\} \tag{3}$$

where C_T is the class of the maximum matching degree with regard to the antecedent fuzzy sets, forming a candidate if-then rule relating the antecedents and the class.

3. Set the rule weight w_j to a candidate rule with the following value if its class C_T is the unique one that takes the maximum matching degree in Eqn. (3):

$$w_j = (\beta_{C_T} - \beta)/\sum_{h=1}^{M} \beta_{C_h} \tag{4}$$

$$\beta = \sum_{C_h \neq C_T} \beta_{C_h}/(M-1) \tag{5}$$

where β is the sum of the matching degrees for all training patterns belonging to the same fuzzy subspace, except those covered by C_T. Otherwise, discard the corresponding candidate rule when two or more classes take the maximum value in Eqn. (3) or all the β_{C_T} are zero, since it cannot be uniquely determined or there is no training pattern in support of this rule.

4. Promote all remaining candidate rules as the members of the learnt rule base, with their corresponding rule weights assigned.

Note that the above method for rule generation and rule weight specification is straightforward when a two-class problem is considered. For instance, assuming that $\beta_{C_1} > \beta_{C_2}$, the consequent class is determined to be Class 1 and its weight will be $(\beta_{C_1} - \beta_{C_2})/(\beta_{C_1} + \beta_{C_2})$. Interestingly, suppose that there are only very few Class 2 patterns in the training data set, the result will be $\beta_{C_1} >> \beta_{C_2} \approx 0$ and $w_j \approx 1$. If however, the total matching degrees of patterns for Class 1 and Class 2 are very similar to each other $\beta_{C_1} \approx \beta_{C_2}$, then $w_j \approx 0$.

2.2 Fuzzy Rule Firing

A popular and easy to understand, and perhaps also the simplest method for classifying a new pattern is based on the strategy of "single winner rule" or "winner taking all" [32]. This is employed in this work (but if preferred, others can be used alternatively which can be found in [33]). The class C_{X_p} of pattern X_p is determined by

$$C_{X_p} = \underset{C_h, h=1,2,...M}{\arg\max}\ \alpha_{C_h} \tag{6}$$

where α_{C_h} is

$$\alpha_{C_h} = \max\{(\prod_{i=1}^{n} \mu_{A_{ji}}(x_{pi}))w_j | w_j \text{ is associated with } R_j, R_j \text{ is associated with } C_h, j = 1, 2, ..., N\} \tag{7}$$

The inferred class is the consequent of the fuzzy rule that has the maximum value of antecedent matching degree by the corresponding rule weight. If two or more classes take the maximum value in Eqn. (6) or the matching degree is zero at X_p, then the pattern cannot be uniquely classified. To force a classification (if desired), such a pattern may be assigned with a default class label that is associated with the most training instances.

2.3 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) was first introduced in [25], and was intended for simulating the flocking and schooling patterns of birds and fish. PSO is a meta-heuristic population-based algorithm, and has been successfully applied to various applications (e.g., [20, 22]). PSO optimises a problem with a population of particles representing candidate solutions. These candidate solutions are updated stochastically with a guide towards the previously best known positions in the search space.

Two primary operations are involved in particular, for the update of PSO processes: velocity update and position update. During each updating iteration, termed generation, every particle's movement is influenced by its local position as well as by the currently known best global position in the search space. A new velocity vector is then computed for each particle based on its current velocity, the distance from its previous best position, and the distance from the global best position so far. The new velocity is in turn used to calculate the next position for each particle in the search space.

More formally, the velocity update for each generation is implemented through the following assignment:

$$v_x = wv_x + c_1 r_1(x_{gBest} - x) + c_2 r_2(x_{pBest} - x) \tag{8}$$

where w is the so-called inertia weight that affects the trade-off between convergence and exploration-exploitation in the PSO updating process; c_1 and c_2 are two positive constants, termed social and cognitive scaling parameter in the literature, respectively; r_1 and r_2 are

two random numbers within the range [0, 1], introducing the stochastic nature during the update; x is the position of a certain particle dimension (or the fitness of the rule weight of a certain rule that leads to the current classification accuracy, in terms of the present application problem); x_{gBest} is the global best position of all particles (namely the fitness of the rule weights currently capable of achieving the highest classification accuracy overall); and x_{pBest} is the best individual position where the particular particle p achieves the current best position. The position is itself updated by the assignment:

$$x = x + \epsilon v_x \tag{9}$$

where ϵ is a further real-valued parameter that is used to control the evolving speed. The interaction between PSO positions and PSO velocities is illustrated in Fig. 2.

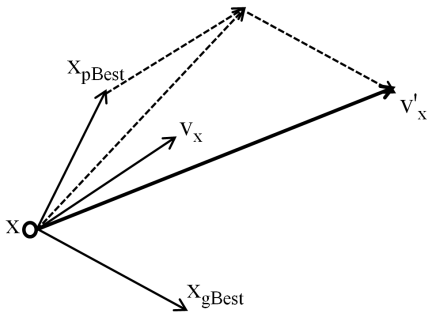

Fig. 2 Update of PSO velocity and position

Both the global best position and the best individual position are used during the update process, with the swarm collectively moving towards the overall best position. The process is iterated for a set of times or until a minimal error is achieved. The overall PSO process is summarised as shown in Algorithm 1.

3 Rule Weight Refinement with PSO

This section first illustrates how the classification boundary may be affected with a set of rule weights taking different values, reinforcing the need for the development of the current work. It then introduces how PSO is employed to refine rule weights for FRBCSs, followed by a description of the general structure of the present work, including a brief analysis of the algorithm complexity.

Algorithm 1 PSO update process

```
for each particle do
    Initialising particle
end for
while maximum iteration or minimum error not attained do
    for each particle do
        Calculating fitness value
        if fitness value is better than pBest then
            Set pBest = current fitness value
        end if
        if pBest is better than gBest then
            Set gBest = pBest
        end if
    end for
    for each particle do
        Calculating particle velocity according to velocity update equation (8)
        Calculating particle position according to position update equation (9)
    end for
end while
```

3.1 Influence of Rule Weights upon Classification Boundaries

A simple example will help demonstrate the effects of adjusting rules weights on the accuracy of the resulting classification. Consider the following case with a two-dimensional input space. For each of the two input variables, x_{p1} and x_{p2}, suppose that three descriptive fuzzy sets are defined such that x_{p1} may take a value on either $A_{11} = Small$, $A_{12} = Medium$, or $A_{13} = Large$, and x_{p2} on either $A_{21} = Short$, $A_{22} = Medium$, or $A_{23} = Long$. The two-dimensional pattern space is then divided into $3^2 = 9$ fuzzy subspaces, as shown in Fig. 3. Each input subspace forms a possible fuzzy if-then rule. The dotted lines in Fig. 3 also show the classification boundaries.

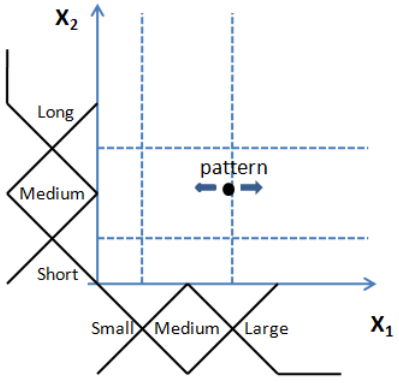

Fig. 3 Fuzzy subspace of a two-dimensional pattern space

A newly collected pattern X_p is classified by first fuzzifying the attribute values using the given prede-

fined membership functions, and then checking if there is any match between the fuzzified value and the antecedent of each given rule. Following the "single winner rule" strategy, the pattern is deemed to be of the class that is associated with the rule which has the following maximum matching degree:

$$C_{X_p} = \underset{C_h, h=1,2,\ldots M}{\arg\max} \gamma_{C_h} \tag{10}$$

where γ_{C_h} is of the same value as α_{C_h} that is obtained from Eqn. 7 when the value of every rule weight is set to 1. This is generally depicted in Fig. 4 (adapted from [21]), where $\alpha(X_p, R_{hj})$ stands for the matching degree of the pattern X_p and the subspace of which is covered by those rules whose consequent is C_h.

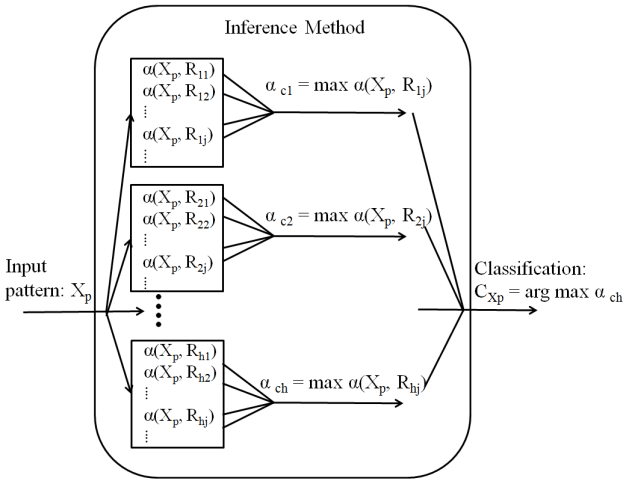

Fig. 4 Single winner rule

When there are patterns misclassified, classification boundaries could be adjusted to recover the classifier performance by modifying the membership functions of the linguistic values. Figure 5 shows a case where the classification boundary is adjusted by modifying the membership functions of fuzzy sets on x_1 axis. Although modifying potential membership functions can adjust the classification boundary and improve the performance of a fuzzy rule-based system [10], it may destroy the potential linguistic meanings given by the domain experts and hence, the interpretability of the learnt model.

According to Eqn. (7), the class label for a new pattern X_p is determined by both the matching degree of its fuzzified value with the antecedent of a rule and its corresponding rule weight. It is possible for a pattern to be misclassified. This is because a pattern may fall into one of the different neighbouring classes implied by several adjacent fuzzy rules, as shown in Fig. 3 where

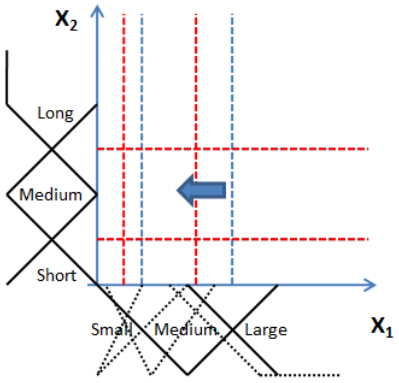

Fig. 5 Modification of classification boundary on membership functions

the black dot is on the edge of two fuzzy subspaces. For a two-dimensional problem, for instance, the equation $\mu_{A_j}(X_p)w_j = \mu_{A_{j'}}(X_p)w_{j'}$ holds while deciding on which class a given pattern may belong to. This observation indicates that the classification boundary can be determined by the ratio of w_j and $w_{j'}$ only. Consequently, the areas dictated by any two neighbouring classification rules may be linearly expanded or narrowed by the ratio of their rule weights. Consider rules R_{j1}, R_{j2}, and R_{j3} as an example in Fig. 6. Instead of modifying membership functions, keeping the rule weights of w_{j1}, w_{j3} unchanged but reducing the value of w_{j2}, the areas covered by R_{j2}'s neighbouring rules R_{j1} and R_{j3} will be expanded while the area covered by R_{j2} is contracted.

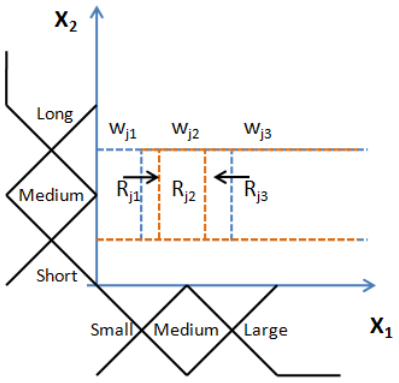

Fig. 6 Modification of classification boundary by adjusting rule weights

Rule weights dictate the exact decision areas originally depicted by the predefined fuzzy sets [27]. Furthermore, the closer the value of a rule weight is to 1, the more reliable or more significant the rule is. With "single winner rule" as the reasoning strategy, any modification of rule weights, through increasing or reducing the weight value, is in fact equivalent to adjusting the

reliability of the relevant individual rules. This is in turn equivalent to reshaping the overall classification boundaries. The adjustment of any two neighbouring rule weights is linear in determining new classification boundaries, but the situation will become much more complicated if the modification of all rule weights is performed simultaneously. Figure 7 shows an example of one possible irregular classification boundary which involves various rules weights [17].

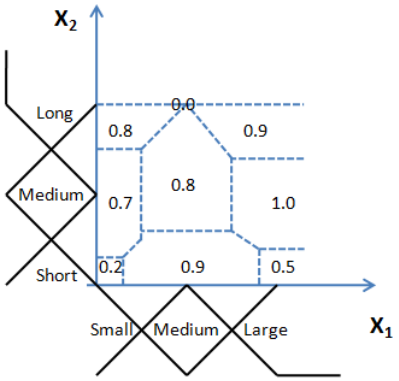

Fig. 7 Classification boundary of an irregular shape

In order to obtain a higher classification accuracy the rule weight associated with the (desirable) wining rule may need to be increased. However, adjusting the rule weight for any individual rule also affects the classification boundaries of its neighbouring rules. That is, whilst the performance of a certain fuzzy rule may be improved by directly changing its rule weight, the performance of its neighbouring fuzzy rules may be deteriorated as a consequence. The overall consequence is thus unpredictable when all the fuzzy rules are changing successively. A method is therefore required to deal with all existing rule weights in a synchronised manner to achieve overall optimal classification performance.

Broadly speaking, the process of finding an optimal combination of a full set of rule weights appears similar to the behaviour of a particle swarm going towards the best solution with each particle's movement influenced by both its local best position and the currently best known position amongst all rules, as with typical applications of Particle Swarm Optimisation (PSO) [25]. Inspired by this observation and the success of PSO in obtaining optimal solutions in multi-dimensional search space, PSO is employed below to evolve the weights associated with a set of fuzzy rules.

3.2 Rule Weight Refinement with PSO

Further to the power of searching for optimal solutions in a discrete search space, PSO can also deal with real numbers directly (and hence the term optimisation rather than search is used). In particular, through encoding rule weights with real numbers, the mechanisms of updating particle velocities and positions, which are inherent to PSO (as reflected by Eqn's. 8 and 9), help make straightforward modifications on the rule weights. The dimensionality that each particle can have is herein set to be the same as the number of the input variables considered in the problem. In utilising PSO for tuning the rule weights, the PSO only needs to maintain a single static population whose members are modified in response to new discoveries about the search space. Each particle typically starts at a random location [34], and is accelerated during the iterations towards the particles that have achieved the previous best position and the global best position so far. The position of a particle corresponds to the fitness measure that determines the quality of the emerging solution.

In the present work, an initial fuzzy if-then rule base is firstly built with a number of predefined fuzzy sets, each having a predefined meaning given by domain experts. This is done via the use of Eqn's. (2) and (3) first, in an effort to obtain a consequent class for a certain rule and then, Eqn's. (2), (4) and (5) are utilised to create initial rule weights for the resulting rules. Tuning the rule weights is regarded as an optimisation problem of concurrently finding the best combination of the weights being amended.

To obtain an optimal set of rule weights with PSO, the problem needs to be interpreted in terms of PSO specification. In particular, each of the existing weights is encoded as one particle dimension, and one particle then represents the entire set of the rule weights associated with the existing fuzzy if-then rules. Positions of the particles in the first generation are initialised with the rule weights obtained by the use of Eqn's. (2), (4) and (5). Particles are then iteratively modified towards the best solution with regard to a given quality measure over the rule weights. The fitness function of each particle is herein gauged by the classification accuracy that is entailed by the renewed fuzzy if-then rules. In summary, the algorithm using PSO to evolve the rule weights of an existing fuzzy classification system is presented in Algorithm 2, supported by Algorithm 3.

Algorithm 2 Fuzzy rule refinement

MAX_IT : number of maximum iterations;
$GOAL$: desired fitness value.

```
Initialisation
repeat
    for each particle i ∈ S do
        if f(x_i) < f(pBest_i) then
            pBest_i = x_i
        end if
        if f(pBest_i) < f(gBest) then
            gBest = pBest_i
        end if
    end for
    for each particle i ∈ S do
        for each dimension d ∈ D do
            v_{i,d} = w v_x + c1 r1 (x_{gBest} − x) + c2 r2 (x_{pBest} − x)
            x_{i,d} = x + ε v_{i,d}
        end for
    end for
    it++
until it > MAX_IT or GOAL is achieved
```

Algorithm 3 Initialisation of fuzzy rule refinement

S : number of particles;
D : number of dimensions equal to number of rules;
r_d : weight associated with rule d in rule base;
$f()$: fitness function used to evaluate particles.

```
for each particle i ∈ S do
    for each dimension d ∈ D do
        x_{i,d} = r_d
        v_{i,d} = Rnd(−v_max/3, v_max/3)
    end for
    pBest_i = x_i
    if f(pBest_i) < f(gBest) then
        gBest = pBest_i
    end if
end for
```

3.3 Learning Classifiers with PSO Refined Rule Weights

As a summary, Fig. 8 shows the general framework of the proposed approach, for situations where the interpretability of fuzzy sets pre-defined by domain experts is required to remain unchanged. The initial rule base can be obtained by simple fuzzy grid partitioning [26] or other data-driven based methods [7,35]. Specification of the initial rule weights can be obtained from a range of methods [18]. PSO is then directed to modify the rule weights, aiming at improving the overall performance of the fuzzy classifier under consideration.

In terms of core algorithm complexity, during each PSO iteration a given set of rules, each of which is associated with an updated rule weight (which may remain the same as its original), is re-evaluated with regard to a global best set of weights achieved so far. Every training data is checked against each fuzzy rule that is associated with the updated rule weight, in order to determine its classification result by the use of single winning rule strategy. The total computation effort required to accomplish re-evaluation is therefore in proportion to the product of the number of training data by the number of fuzzy rules, denoted as m and N respectively, namely $O(mN)$.

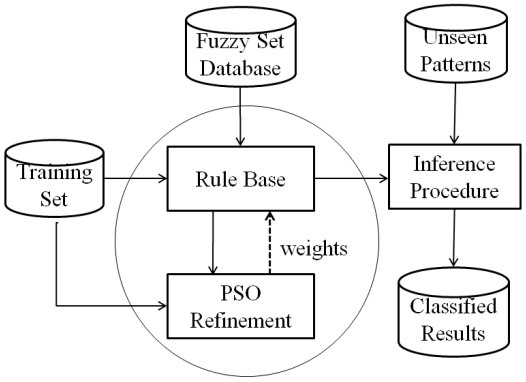

Fig. 8 Framework of FRBCS with PSO refined rule weights

Obviously, in developing an FRBCS this way, a training data set is needed as input to the learning system, for both the generation of the initial rule base and the process of the rule weight refinement. For a given training set, the greater the number of initially built fuzzy rules, the more computation is needed to complete a PSO update process. The training of the FRBCS completes once the PSO-based refinement process terminates. Unseen patterns can then be classified by the trained classifier. Although the single winner rule strategy is adopted to classify patterns here, other inference methods (e.g., weighted vote) may also be employed if preferred [33]. Note that if after a training process is completed, a newly collected set of data becomes available then this set can also be utilised to train the classifier, with new rules integrated into the existing rule base. The application of this idea would make the resulting FRBCSs dynamically adaptive, but how this may be implemented with minimal disruption of the existing rule base remains as further work.

4 Experimental Results

To demonstrate the potential of the proposed approach, a number of comparative experiments are carried out. The results are reported and discussed here, in terms of the effects of: (a) rule weighting schemes, (b) rule base sizes, and (c) rule learning methods used.

4.1 Experimental Setup

Experiments are performed on 12 real-valued UCI benchmark data sets [36]. A summary of the characteristics of these data sets is given in Table 1. The PSO parameters are empirically specified in Table 2. Note that similar settings can be found in the relevant work existing in the literature (e.g., [23]), supported by the insights gained through a survey on PSO parameter selection as reported in [37]. Note also that as the main aim of this study is to examine the efficacy of applying PSO for fuzzy rule refinement instead of that of PSO itself, only the basic version of PSO is used in the experiments. The parameter specification for PSO is not carefully adjusted, therefore, simulation results could be further improved where more sophisticated versions of PSO are used with carefully modified parameters.

Table 1 Summary of data sets used

Data Set	Attributes	Classes	Instances
ecoli	7	8	336
glass	9	7	214
haberman	3	2	306
image-segmentation (training)	19	7	210
iris	4	3	150
liver-disorders	6	2	345
new-thyroid	5	3	215
parkinsons	22	2	195
pima-diabetes	8	2	768
prnn-synth	2	2	250
seeds	7	3	210
yeast	8	10	1484

Initial rule weights are calculated via Eqn's. (2), (4) and (5), classification accuracies are computed with or without any initial heuristically produced rule weights, in order to show how the rule weight refinement may affect the performance of the learned rules' accuracy. In Table 3, the abbreviations PSO-FR, FR, and H-FR stand for the application of fuzzy rules with PSO-refined rule weights, that of fuzzy rules without rule weights, and that of fuzzy rules with heuristic rule weights initially provided, respectively. The purpose of this experimental design is to test how additional rule weight may affect the performance of a potential classifier, and how the proposed method may help improve such performance. Note that several popular rule-based learning classifiers are also selected to support the comparative study. This is to demonstrate that simple FRBCSs which employ a rule base whose individual rule weights are modified with a PSO process are competitive in their performance as with popular rule-based classifiers available in the literature.

Table 2 Empirical PSO parameter settings

w	c_1	c_2	ϵ	$Max\_Generation$	$Particle\_Numbers$
0.8	2.0	2.0	1.0	200	30

In an effort to examine the effect of using PSO-refined rule weights upon the improvement of fuzzy partition quality, four different fuzzy partitions are tested, where each of the pattern spaces is divided into K ($K = 2, 3, 4, 5$) triangular fuzzy subsets in the same way as that shown in Fig. 1. This allows the performance of the proposed method to be investigated for fine fuzzy partitions as well as coarse fuzzy partitions. In particular, the case of $K = 2$ represents a very rough partition, while that of $K = 5$ represents a very detailed partition. Similar partitions can be found in [13]. Note that given a K, in theory, the total number of fuzzy if-then rules for each fuzzy partition would be K^n, where n stands for the number of input attributes, however a fuzzy rule will only be generated when there is a training pattern covered by an emerging rule. So, the total number of rules produced is typically much smaller.

Owing to a large amount of systematic experimental investigation being carried out, only stratified twofold cross-validation (2-CV) is employed for data validation in this work. In 2-CV, a given data set is partitioned into 2 subsets. One of the subsets is used to train a fuzzy classifier, where the proposed approach is used to refine corresponding fuzzy rule weights. Another divided subset is retained as the testing data to produce a single accuracy value. The process is then repeated 30 times by initialising different, randomly assigned seeds to produce the final average outcomes. Pairwise t-tests are run with $p < 0.05$. Results are thus measured in terms of the significance of differences between different learning classifiers, with the achieved accuracy of PSO-FR as the reference in each experiment. Those results that are significantly better, worse or of no difference are marked with "(v)", "$(*)$", or "$(-)$", respectively.

4.2 Effect of Rule Weighting Scheme

As shown in Table 3, H-FR outperforms FR in terms of average classification accuracy, regardless of the number of fuzzy partitions, for 7 out of 12 data sets (including: ecoli, iris, image, liver-disorders, new-thyroid, parkinsons, and prnn-synth). For the other 5 data sets, the results of H-FR are competitive to those of FR. This is not surprising since the rule weights used in H-FR are heuristically initialised. This conforms to what

is discussed in Section 3.1 regarding the influence of rule weights upon classification boundaries.

Although H-FR generally achieves better results than FR, the performance of H-FR is still far from ideal. Fortunately, as illustrated in Table 3, the results of PSO-FR are significantly better than those achievable by H-FR for 33 times and worse for just once, with 14 ties. This superior performance of learnt fuzzy classifiers with PSO refined rule weights is reinforced by Fig. 9, which systematically depicts the relation between the PSO iteration number and the accuracy of a learnt classifier for each of the simulated data sets. In this figure, 12 sets of plots are shown each representing the results on one data set for both training and testing performance using 4 different fuzzy partitions, namely $K = 2, 3, 4, 5$. Generally, for both training and testing data, each FRBCS with the current PSO-returned rule weights starts from their initial performance, through an oscillatory process, and then reaches a steady state with a noticeable degree of improvement.

4.3 Effect of Rule Base Size

From Fig. 9, further observations can be obtained. For better viewing, the accuracy of each classifier is displayed for every 3 iterations within a total of 100 iterations, each point is the average of the results from 30 runs of 2-CV. As can be seen, after an initial period of oscillations, generally the trend of the training performance for all FRBCSs tends to converge at around $40th$-$60th$ iteration regardless of the number of fuzzy partitions. In terms of testing accuracies, the curves are generally more oscillatory than the training ones. Although the testing accuracies do not reach so high as that is achievable over the training phase, they are significantly improved over the original performance.

Examining these plots more carefully, it is interesting to note that in general, fuzzy classifiers modelled with a lower number of partitions tend to have a poor performance at the beginning for both training and testing, likely due to the coarse partitioning of the input spaces. However, in terms of testing curves, although coarse partitioned ones (e.g., $K = 2$) have a lower start, their performance can outperform the finer partitioned ones (e.g., $K = 5$), not just catching up with them, particularly for diabetes, glass, haberman, liver-disorders, parkinsons, prnn-synth, seeds, and thyroids (8 out of 12 data sets). For finer partitions, which generally have a better start in performance, the classification accuracy does not improve so much as lower partitioned ones when converged, and even underperformed than those models with a lower number of partitions in 11 out of 12 data sets (including: diabetes, ecoli, glass, iris, image, liver-disorders, parkinsons, prnn-synth, seeds, thyroid, and yeast). Although a fuzzy classifier using a finer partition and hence, with more initial rules is likely to have a head start regarding classification performance, the resulting more complicated search space may make final solutions converge in a local minimal, thereby achieving worse results than those obtainable by the use of coarsely partitioned ones. This hints that in practical applications of the present work, not too large a K value should be employed.

In generating the initial rule base by the use of fuzzy grid partitioning, finer partitions of the input spaces lead to more fuzzy rules, as clearly indicated from the rule numbers in Table 3. The more fuzzy rules are generated initially, the more rule weights need to be modified. This implies that the search space becomes larger, the PSO process involves higher computational complexity and also, the classification results are less interpretable. Besides, as observed above, a finer partitioned fuzzy classifier normally achieves worse performance. One possible reason for such seemingly unintuitive results is overfitting during the training. Therefore, it would be worthwhile to consider the number of fuzzy rules as part of the criteria in constructing the fitness function, by penalising emerging models that consist of more rules or by filtering poor quality individual rules (e.g., low coverage or low performance). The implementation of such ideas remains as further research.

4.4 Effect of Rule Learning Method

Three classifier learning algorithms that generate models in the form of a rule set are chosen to perform classification tasks for comparison purpose. These are: the popular C4.5 decision tree learner (J48) [29], the top-down fuzzy pattern trees (PTTD) [30], and the fuzzy subsethood-based rule models with quantifiers (QSBA) [31].

Fuzzy pattern tree induction is recently introduced as a novel machine learning method for classification [38]. A pattern-tree classifier is composed of an ensemble of pattern trees, each of which is of a hierarchical structure, whose inner nodes are marked with generalised (fuzzy) logical and arithmetic operators, and whose leaf nodes are associated with fuzzy predicates applied to the input variables [30]. In order to reduce the runtime of PTTD and to have a fair comparison, only the algebraic t-norm and maximum s-norm are chosen as fuzzy operators in this work, which are similar to the operators used in the proposed approach herein (see Eqn's. (6) and (7)). Fuzzy quantifier-based models are generated using fuzzy quantification to replace crisp weights in subsethood-based fuzzy rule models, which

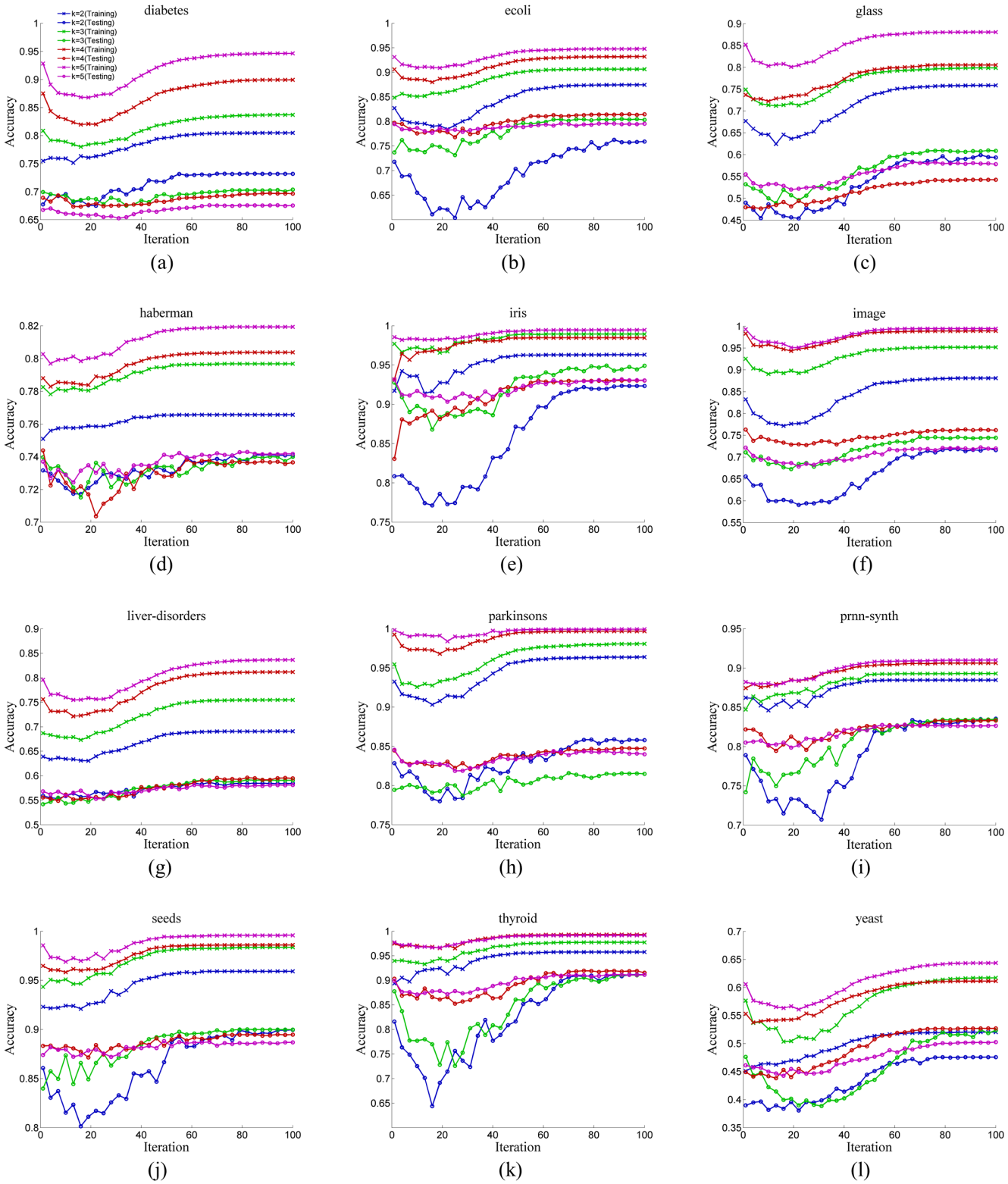

Fig. 9 Relation between PSO iteration number and classification performance

Table 3 Comparison using 30 × 2 cross-validation with respect to classification accuracy (%), where v, $-$ or $*$ indicate statistically better, same or worse results, respectively, and bold figures signify overall best results for each data set with a certain partition number.

Data Sets	K	Rule Number	PSO-FR	FR	H-FR	J48	PTTD	QSBA
ecoli	2	25.97	**78.28**±1.97	72.83±1.29(*)	75.98±1.65(*)	74.97±1.29(*)	76.21±2.19(*)	23.29±6.54(*)
	3	39.30	**80.49**±2.10	72.21±1.39(*)	74.99±1.49(*)	77.40±1.93(*)	76.24±1.48(*)	19.59±7.86(*)
	4	56.58	**81.53**±1.75	80.78±1.86(*)	81.47±1.51(-)	75.34±2.40(*)	78.53±2.02(*)	58.64±3.09(*)
	5	84.42	79.58±1.84	79.04±2.37(*)	**79.65**±2.18(-)	75.65±1.90(*)	77.97±1.92(*)	69.03±3.19(*)
glass	2	23.98	**60.67**±4.77	49.42±3.22(*)	52.23±4.11(*)	52.13±2.62(*)	59.14±2.88(*)	28.30±4.56(*)
	3	31.48	61.12±2.50	57.90±2.40(*)	55.51±4.21(*)	59.03±3.11(*)	**61.57**±4.01(*)	36.08±3.88(*)
	4	40.82	54.25±4.41	48.33±3.44(*)	49.10±3.37(*)	57.99±3.37(v)	**59.31**±4.00(v)	37.88±3.90(*)
	5	56.70	58.07±3.03	54.31±2.95(*)	58.47±2.91(-)	57.54±3.89(-)	**63.93**±3.63(v)	45.83±4.34(*)
haberman	2	3.57	**74.07**±1.07	73.10±0.27(*)	73.27±0.43(*)	73.35±0.48(*)	72.41±1.63(*)	72.57±4.23(*)
	3	6.50	74.02±1.47	73.28±1.05(*)	73.14±0.74(*)	73.33±0.67(*)	73.35±1.43(*)	**74.32**±1.16(-)
	4	8.87	73.65±1.39	**75.52**±1.17(v)	74.18± 1.10(v)	73.24±0.67(-)	74.90±1.38(v)	73.77±2.87(-)
	5	13.47	**74.18**±1.64	73.33±1.51(*)	73.77±1.39(*)	73.16±0.80(*)	73.81 ±1.08(-)	73.29±1.29(*)
image-segmentation	2	37.05	72.49±3.33	69.41±3.53(*)	70.37±3.23(*)	**74.78**±2.13(v)	64.98±3.92(*)	55.32±1.55(*)
	3	65.15	74.68±2.38	70.86±2.59(*)	73.54±2.45(*)	76.49±2.70(v)	**80.98**±2.57(v)	59.14±6.81(*)
	4	86.60	76.44±2.74	76.57±2.89 (-)	76.57±2.74(-)	**82.70**±2.28(v)	80.97±2.37(v)	72.09±7.56(*)
	5	93.07	72.41±2.11	72.40±2.06(-)	72.51±2.11(-)	80.46 ±2.53(v)	**83.68**±2.15(v)	74.45±6.92(-)
iris	2	7.98	**92.33**±2.90	72.04±2.00(*)	84.58±2.64(*)	76.65±2.76(*)	77.49±2.27(*)	66.67±0.00(*)
	3	14.75	95.16±1.60	91.56±1.37(*)	93.89±0.91(*)	**95.33**±1.19(-)	92.18±0.89(*)	62.11±1.63(*)
	4	22.38	**93.02**±1.93	78.18±2.36(*)	85.60±2.75(*)	90.09±3.12(*)	90.78±3.80(*)	62.11±1.71(*)
	5	30.60	93.09±1.66	93.00±1.33(-)	93.22±0.89(-)	91.53±2.37(*)	94.73 ±0.98(v)	**94.91**±0.93(v)
liver-disorders	2	13.97	**58.45**±2.07	56.10±1.40(*)	57.72±1.53(*)	56.98±1.62(*)	58.20±2.01(-)	47.18±3.03(*)
	3	35.80	59.07±2.89	52.10±2.59(*)	56.45±2.41(*)	56.91±1.40(*)	**59.71**±3.16(-)	46.07±1.54(*)
	4	56.53	59.52±3.08	54.64±2.93(*)	56.26±2.80(*)	56.25±2.26(*)	**60.03**±2.55(-)	53.07±3.22(*)
	5	82.30	58.14±2.67	56.24±2.55(*)	56.75±1.91(*)	56.11±2.34(*)	**63.51**±2.78(-)	57.92±3.33(-)
new-thyroid	2	6.87	**91.18**±1.49	83.97±1.19(*)	85.13±1.26(*)	84.85±1.79(*)	83.71±0.68(*)	87.21±2.76(*)
	3	16.88	**91.13**±2.31	88.34±1.40(*)	89.30±1.41(*)	86.25±1.62(*)	87.06±1.09(*)	89.71±2.61(*)
	4	25.78	91.92±1.77	90.32±1.54 (*)	91.60±1.03(-)	88.50±2.41(*)	92.34±0.87(-)	**93.38**±0.70(v)
	5	33.33	91.16±1.44	88.65±1.95(*)	91.09±1.34(-)	90.90±1.56 (-)	89.61±1.35(*)	**92.67**±0.80(v)
parkinsons	2	57.07	86.10±2.19	79.15±2.12(*)	83.85±2.29(*)	82.37±2.47(*)	**86.19**±1.35(-)	54.75±5.06(*)
	3	79.67	81.47±2.45	79.83±2.11(*)	80.72±2.48(*)	**86.61**±2.01(v)	83.72±1.44(v)	77.09±0.86(*)
	4	88.12	84.74±3.27	84.80±2.82(-)	84.86 ±2.92(-)	84.35±2.89(-)	**85.39**±1.87(-)	76.97±0.90(*)
	5	93.38	84.78±3.77	84.71±3.80(-)	84.75±3.77(-)	84.48±2.02(-)	**86.46**±1.87(v)	77.88±1.74(*)
pima-diabetes	2	36.32	**73.19**±1.57	66.57±1.13(*)	69.34±0.92(*)	68.09±1.28(*)	69.65±1.40(*)	70.53±0.66(*)
	3	84.08	70.30 ±1.38	70.72±1.56(-)	69.83±1.41(*)	72.70±1.00(v)	**73.83**±0.44(v)	61.52±1.60(*)
	4	201.70	69.67±1.66	68.14±1.34(*)	69.51±1.60(-)	**73.94**±0.82(v)	71.85±1.66(v)	58.29±1.34(*)
	5	269.70	67.59±1.86	66.65±1.85(*)	67.50±1.86(-)	**74.40**±1.04(v)	73.55±1.05(v)	69.05±0.71(*)
prnn-synth	2	4.00	**83.67**±1.83	80.97±0.20(*)	81.25±1.54(*)	81.83±1.12(*)	81.83±1.12(*)	51.68±1.99(*)
	3	7.90	**83.45**±1.29	70.08±3.46(*)	80.47±1.13(*)	77.03±2.19(*)	72.04±2.47(*)	71.20±3.40(*)
	4	12.52	83.32±2.21	82.28±0.91(*)	84.23±1.15 (v)	83.28±1.78(-)	**84.24**±1.21(v)	83.19±1.29(-)
	5	16.52	82.65±1.63	79.36±2.35(*)	83.29±1.21(v)	82.76±1.52(-)	80.60±2.12(*)	**83.52**±1.04(v)
seeds	2	16.22	90.06±1.86	88.21±0.98(*)	86.49±1.26(*)	87.33±1.79(*)	**92.16**±1.34(v)	33.39±0.35(*)
	3	41.92	**89.98**±1.83	79.67±1.55(*)	85.95±2.26(*)	84.44±2.06(*)	86.35±1.55(*)	41.13±1.27(*)
	4	56.17	**89.49**±1.74	88.08±1.24(*)	88.95±1.38(*)	87.84±2.47(*)	88.35±1.55(*)	62.22±1.45(*)
	5	75.57	88.70±1.61	87.89±1.71(*)	88.06±1.67(*)	86.94±1.56(*)	**88.78**±1.52(-)	73.08±1.61(*)
yeast	2	34.40	47.58±1.84	38.49±0.51(*)	38.79±2.53(*)	39.57±1.79(*)	**50.78**±0.92(v)	14.22±3.87(*)
	3	83.73	**54.90**±0.95	51.56±0.66(*)	53.17±0.97(*)	52.34±1.49(*)	50.63±1.67(*)	10.45±2.15(*)
	4	90.42	**52.70**±1.16	42.05±0.92(*)	49.76±1.23(*)	51.32±1.68(*)	52.19±1.21(-)	32.53±2.36(*)
	5	164.65	50.33±1.45	47.31±1.29(*)	48.30±1.04(*)	49.42±1.49(*)	**51.90**±1.98(v)	46.89±1.66(*)

are not only interpretable but also practically applicable [39], [40]. Note that the same fuzzy pre-partition of the input space is adopted for both PTTD and QSBA as that for the proposed method, whereas the same partitioning interval is chosen as the corresponding variable discretisation for J48. All these algorithms are implemented within the WEKA machine learning framework [41] with default parameter setting unless otherwise stated previously.

The winning results in terms of achieving the highest classification accuracy per learning classifiers are highlighted in boldface in Table 3. Note that the proposed method (PSO-FR) has 18 wins, compared with 17 wins by PTTD, 6 wins by J48, and 5 wins by QSBA. Obviously the proposed approach significantly outperforms J48 and QSBA, and is competitive to PTTD. Between the two better performers, PSO-FR and PTTD, a specific comparison can be made from the results obtained. Statistically, the proposed method wins 22 times and loses 16 times with 10 ties over PTTD. These results jointly demonstrate that the present work is at least competitive to the state-of-the-art rule-based classifiers in the literature regarding classification accuracy.

5 Conclusion and Future Work

This paper has proposed an approach for fuzzy rule weight refinement by the use of PSO. The approach works for situations where an initial rule fuzzy rule-base has been built with predefined fuzzy sets, which are required to be maintained for the purpose of consistent interpretability, both in the learned models and in the inference results using such models. Systematic experimental results have demonstrated the following:

1. The performance of a fuzzy rule-based classifier can be significantly improved with rule weight refinement implemented by PSO.
2. The size of an initially built rule base may affect the performance of the proposed method (but optimisation of the initial fuzzy quality space is expected to help reduce such influence).
3. The approach is at least competitive to typical state-of-the-art learning classifiers even when only simple fuzzy grid partitioning is used to create the initial rule base.

Whilst promising, work remains to further improve this approach. In particular, currently, only the accuracy of a fuzzy learning classifier is considered as the criterion or fitness measure when evolving rule weights. However, as the size of the rule base or equivalently the number of rule weights will affect the final result, the number of rules and hence the partition of the input space should become part of the fitness function. The optimisation of PSO parameters also needs to be examined in order to strengthen the ability of the proposed method since the current implementation does not investigate such potential effects. Furthermore, instead of using PSO, it would be interesting to see whether the use of an alternative evolutionary computation mechanism may help develop better fuzzy learning classifiers, regarding both effectiveness and efficiency.

Finally, note that this work assumes the availability of an initial rule base, be it created by a simple partitioning scheme as used in the present implementation, given by the human expert, or generated by a certain rule learning method automatically (e.g., through a clustering algorithm like fuzzy c-means). It investigates the efficacy of modifying rule weights for rules given in such an initial rule base. The modification process itself has little to do with the set-up of the initial rule base, which is fixed throughout. However, in the extreme cases where an exceedingly fine-tuned initial rule base is available such that the rules are already too accurate to be improved, the need for weight adjustment may vanish. Therefore, it is worth examining the efficacy of the proposed approach when different rule bases initialisations are involved, including the situations where different variables have different partitions of their respective underlying domains. This remains active research.

6 Acknowledgements

The first and third authors are grateful to Aberystwyth University for providing PhD scholarships in support of their research. The authors are also very grateful to the reviewers for their constructive comments which have helped improve this work significantly.

References

1. Y. Yuan and H. Zhuang, "A genetic algorithm for generating fuzzy classification rules," *Fuzzy Sets and Systems*, vol. 84, no. 1, pp. 1–19, 1996.
2. H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, no. 5, pp. 601–618, 1999.
3. Y. Shi, R. Eberhart, and Y. Chen, "Implementation of evolutionary fuzzy systems," *Fuzzy Systems, IEEE Transactions on*, vol. 7, no. 2, pp. 109–119, 1999.
4. C. A. Pena-Reyes and M. Sipper, "A fuzzy-genetic approach to breast cancer diagnosis," *Artificial Intelligence in Medicine*, vol. 17, no. 2, pp. 131–155, 1999.
5. M. Delgado, N. Marín, D. Sánchez, and M.-A. Vila, "Fuzzy association rules: general model and applications," *Fuzzy Systems, IEEE Transactions on*, vol. 11, no. 2, pp. 214–225, 2003.
6. S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework," *Neural Networks, IEEE Transactions on*, vol. 11, no. 3, pp. 748–768, 2000.
7. Q. Shen and A. Chouchoulas, "A rough-fuzzy approach for generating classification rules," *Pattern Recognition*, vol. 35, no. 11, pp. 2425–2438, 2002.
8. R. Jensen, C. Cornelis, and Q. Shen, "Hybrid fuzzy-rough rule induction and feature selection," in *Proceedings of the 18th International Conference on Fuzzy Systems*. IEEE, 2009, pp. 1151–1156.

9. R. Diao and Q. Shen, "A harmony search based approach to hybrid fuzzy-rough rule induction," in *Proceedings of the 21th International Conference on Fuzzy Systems*. IEEE, 2012, pp. 1–8.
10. O. Cordón and F. Herrera, "A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples," *International Journal of Approximate Reasoning*, vol. 17, no. 4, pp. 369–407, 1997.
11. J. G. Marín-Blázquez and Q. Shen, "From approximative to descriptive fuzzy classifiers," *Fuzzy Systems, IEEE Transactions on*, vol. 10, no. 4, pp. 484–497, 2002.
12. D. Nauck and R. Kruse, "How the learning of rule weights affects the interpretability of fuzzy systems," in *Proceedings of the 7th International Conference on Fuzzy Systems*, vol. 2. IEEE, 1998, pp. 1235–1240.
13. M. Z. Jahromi and M. Taheri, "A proposed method for learning rule weights in fuzzy rule-based classification systems," *Fuzzy Sets and Systems*, vol. 159, no. 4, pp. 449–459, 2008.
14. K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy rule-based classification systems," *Fuzzy Systems, IEEE Transactions on*, vol. 4, no. 3, pp. 238–250, 1996.
15. E. G. Mansoori, M. J. Zolghadri, and S. D. Katebi, "A weighting function for improving fuzzy classification systems performance," *Fuzzy Sets and Systems*, vol. 158, no. 5, pp. 583–591, 2007.
16. M. J. Zolghadri and E. G. Mansoori, "Weighting fuzzy classification rules using receiver operating characteristics (roc) analysis," *Information Sciences*, vol. 177, no. 11, pp. 2296–2307, 2007.
17. H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *Fuzzy Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 506–515, 2001.
18. H. Ishibuchi and T. Yamamoto, "Rule weight specification in fuzzy rule-based classification systems," *Fuzzy Systems, IEEE Transactions on*, vol. 13, no. 4, pp. 428–435, 2005.
19. T. Chen, Q. Shen, P. Su, and C. Shang, "Refinement of fuzzy rule weights with particle swarm optimisation," in *Computational Intelligence (UKCI), 2014 14th UK Workshop on*. IEEE, 2014, pp. 1–7.
20. A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.
21. M. Galea and Q. Shen, "Simultaneous ant colony optimization algorithms for learning linguistic fuzzy rules," in *Swarm Intelligence in Data Mining*. Springer, 2006, pp. 75–99.
22. S. Agrawal, K. Panigrahi, and M. K. Tiwari, "Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 5, pp. 529–541, 2008.
23. R. Diao and Q. Shen, "Feature selection with harmony search," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 6, pp. 1509–1523, 2012.
24. P. Su, Y. Li, Y. Li, and S. C.-K. Shiu, "An auto-adaptive convex map generating path-finding algorithm: Genetic convex a*," *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 5, pp. 551–563, 2013.
25. J. Kennedy, R. Eberhart *et al.*, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, no. 2. Perth, Australia, 1995, pp. 1942–1948.
26. H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets and Systems*, vol. 52, no. 1, pp. 21–32, 1992.
27. H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms," *Fuzzy Sets and Systems*, vol. 65, no. 2, pp. 237–253, 1994.
28. H. Ishibuchi, T. Nakashima, and T. Morisawa, "Voting in fuzzy rule-based systems for pattern classification problems," *Fuzzy Sets and Systems*, vol. 103, no. 2, pp. 223–238, 1999.
29. J. R. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.
30. R. Senge and E. Hullermeier, "Top-down induction of fuzzy pattern trees," *Fuzzy Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 241–252, 2011.
31. K. A. Rasmani and Q. Shen, "Modifying weighted fuzzy subsethood-based rule models with fuzzy quantifiers," in *Proceedings of the 13th International Conference on Fuzzy Systems*, vol. 3. IEEE, 2004, pp. 1679–1684.
32. H. Ishibuchi, T. Murata, and I. Türkşen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets and Systems*, vol. 89, no. 2, pp. 135–150, 1997.
33. O. Cordón, M. J. del Jesus, and F. Herrera, "A proposal on reasoning methods in fuzzy rule-based classification systems," *International Journal of Approximate Reasoning*, vol. 20, no. 1, pp. 21–45, 1999.
34. J. Kennedy, J. F. Kennedy, and R. C. Eberhart, *Swarm intelligence*. Morgan Kaufmann, 2001.
35. L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 6, pp. 1414–1427, 1992.
36. K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
37. A. Rezaee Jordehi and J. Jasni, "Parameter selection in particle swarm optimisation: a survey," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 527–542, 2013.
38. Z. Huang, T. D. Gedeon, and M. Nikravesh, "Pattern trees induction: a new machine learning method," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 4, pp. 958–970, 2008.
39. K. Rasmani, J. M. Garibaldi, Q. Shen, and I. O. Ellis, "Linguistic rulesets extracted from a quantifier-based fuzzy classification system," in *Proceedings of the 18th International Conference on Fuzzy Systems*. IEEE, 2009, pp. 1204–1209.
40. K. A. Rasmani and Q. Shen, "Weighted linguistic modelling based on fuzzy subsethood values," in *Proceedings of the 12th International Conference on Fuzzy Systems*, vol. 1. IEEE, 2003, pp. 714–719.
41. I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical machine learning tools and techniques with java implementations," 1999.