A Hybrid of Self Organized Feature Maps and Parallel Genetic Algorithms for Uncertain Knowledge

*[i]Mona Gamal[1] Ahmed Abo El-Fatoh[2] Shereef Barakat[3] Elsayed Radwan.[4]

Mansoura University, Faculty of Computer and Information Sciences
Information System Department[1,2,3]
Assistant Prof., Institute of Scientific Research & Revival of Islamic Culture,
Umm Al-Qura University, KSA. [4]
P.O.Box: 35516

ABSTRACT

The need to handle uncertainty and vagueness in real world becomes a necessity for developing good and efficient systems. Fuzzy rules and their usage in fuzzy systems help too much in solving these problems away from the complications of probability mathematical calculations. Fuzzy rules deals will words and labels instead of values of the variables. These labels are called variable's subsets and needed to be prepared carefully to make sure that the fuzzy rules depend on accurate propositions. This research tries to design an efficient set of rules that is used later for inference by a hybrid model of Self Organized Features Maps and Parallel Genetic Algorithms. Self Organized Features Maps capabilities to cluster inputs using self adoption techniques have been very useful in generating fuzzy membership functions for the subsets of the fuzzy variables. Then the Parallel Genetic Algorithms use these membership functions along with the training data set to find the most fit fuzzy rule set from a number of initial sub populations according to the fitness function. The illustrations of the proposed model and its sub modules along with the experimental results and comparisons with previous techniques in generating rules from data sets are declared.

General Terms

Uncertain Knowledge , Soft Computing , Rule Based Systems.

Keywords

Fuzzy System, Parallel Genetic Algorithms, Self Organized Feature Map.

1. Introduction

Fuzzy system [15] [13] is any system that depends on fuzzy input-output variables instead of regular ones. Unlike regular variables, the values of the fuzzy ones belong to some fuzzy subsets with a degree of membership. One kind of these fuzzy systems is the rule based fuzzy system that decides its output through a set of fuzzy if-then rules by some inference mechanism (inference engine). The fuzzy rules generation is an interesting topic that many researches try to find a model to build them. At first, fuzzy if-then rules were usually derived from human experts but it was a very difficult work to gather these rules from an expert and these rules might be affected by perspective of the expert. Therefore, many approaches were proposed to automatically generate fuzzy if-then rules from training datasets. Theses approaches were interested in the rules accuracy and the speed of generating them. This research tries to combine two machine learning algorithms to generate the fuzzy rules in two stages. The first stage is to prepare the fuzzy variables by generating the membership function for the fuzzy subsets of the variables. This stage is implemented using Self Organized Feature Maps (SOFM) from a previous research. The result of this phase is passed to the Parallel Genetic Algorithm (PGAs) [23] along with the training data to first randomly generate initial subpopulations and then use some fitness function to select the fit rules with respect to the training data. The test data is then used to check for the accuracy of the rules generated.

The rest of this paper is organized as follows: Section 2 is a quick review on the previous research in generating fuzzy rules by evolutionary algorithms. Section 3 represents the preliminaries such as the declaration of the fuzzy systems, Self Organized Feature Maps SOFM and Parallel Genetic Algorithms PGA. Section 4 gives an over view on the whole system and its modules. It goes inside the system to explain in detail the generation of the fuzzy membership functions of the subsets of the fuzzy variables, designing the fuzzy rule set and finding the fitness function for the PGA. Experimental results and conclusion will appear in sections 5 and 6 respectively.

2. Related Work

Genetic Algorithm [4] [19] [22] [27] has been applied for the discovery of fuzzy rules which were competitive to decision tree induction rules from the perspective of predictive accuracy [28]. The individuals of the population were the fuzzy rules to be designed and the final population was the fuzzy rule set. A hybrid algorithm of two fuzzy genetics-based machine learning approaches (i.e., Michigan and Pittsburgh) for designing fuzzy rule-based classification systems has also been proposed [10][11][12]. A new method was also proposed to automatically learn the knowledge base (KB) by finding an appropriate database by means of a genetic algorithm while using a simple generation method to derive the rule base (RB) [3].

As Parallel Genetic Algorithms (PGAs) [23] were used for parallel processing of the fitness function of the individuals, both population and training data set are divided into subgroups. These sub groups are distributed across all available processors, and control parallelism, by distributing the population of individuals across all available processors the fitness evaluation phase exploits both data parallelism and control parallelism by having the individuals passing through all the processors in a kind of round-robin scheme. In this scheme the physical interconnection of processor nodes is

mapped into a logical ring of processor nodes, so that each processor node has a right neighbor and a left neighbor [6]. A different idea is that both a training data set and a population are divided into subgroups (i.e., into training data subsets and sub-populations, respectively) for the use of multiple processors [29]. Other paper proposes genetic-fuzzy approach for discovery of fuzzy decision rules from datasets containing both categorical as well as continuous attributes. The continuous attributes are normalized and fuzzified in pre-processing step. A novel match procedure is devised to take care of mixed attributes during the fitness computations of individual rules. A direct match is carried out for categorical attributes whereas a Mumdani style min-max method is employed for matching continuous attributes with the instances in the training dataset [19].

But these researches assume that fuzzy variables are prepared or use discrimination techniques to make the membership functions of the subsets of the variables. So some researches thought of a technique for estimating the fuzzy subsets. Gene Expression Programming method uses two populations. One for Fuzzy Classification Rules which is evolved by syntax genetic programming and the other one for membership function definitions which is evolved by mutation based evolutionary algorithm. These two populations co-evolve to better classify the underlying data set [1], [25].

Also a new hybrid approach for optimization combining Particle Swarm Optimization (PSO) and Genetic Algorithms (GAs) using Fuzzy Logic for parameter adaptation and to integrate the results. Fuzzy Logic is used to combine the results of the PSO and GA in the best way possible. Also, fuzzy logic is used to adjust parameters in the FPSO and FGA [8] [9]. Other research described the use of Modular Neural Networks (MNN) for pattern recognition in parallel using a cluster of computers with a master-slave topology. Also, a parallel genetic algorithm to optimization architecture was used [7].

3. Preliminaries

3.1 Fuzzy System

Fuzzy System [15] [16] is a system that depend on vague and ambiguous terms (parameters or variables) instead of ordinary variables that have exact values. Generally the fuzzy Systems consist of three main concepts

(1) Fuzzy input and output variables and their fuzzy values.

(2) Fuzzy rules.

(3) Fuzzy inference methods, which may include defuzzification.

Fuzzy input and output variables and their fuzzy values:

Fuzzy variables are variables that have fuzzy values which belong to a set of fuzzy subsets. These fuzzy subsets are called labels, terms or words that define the fuzzy variable. The fuzzy values (elements) of a variable partially belong to the fuzzy subsets. The degree that an element belongs to a fuzzy subset is called the membership degree. This degree of membership is characterized by a fuzzy membership function.

$$\mu_A(u): U \rightarrow [0,1] \qquad (1)$$

where U is called the universe and A is a fuzzy subset of U.

The value of membership are real values in the interval [0,1] where the value 0 means that the element does not belong to the subset A and the variable value 1 means that the value entirely belongs to the subset.

For example figure 1 shows three membership functions representing three fuzzy sets labeled as "short", "medium", and "tall" all of them being fuzzy values of a variable "height". The value 170 cm belongs to the fuzzy set "medium" to a degree of 0.2 and at the same time to the set "tall" to a degree [24].

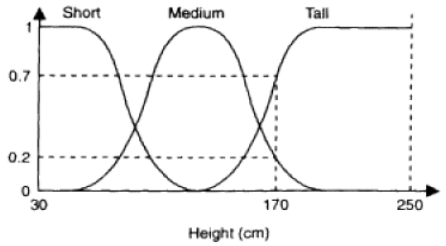

Figure 1[24]: Membership functions representing three fuzzy sets for the variable "height."

Fuzzy rules:

There are many types of fuzzy rules used for building knowledge based fuzzy systems. One of these fuzzy rules is Zadeh-Mamdani's fuzzy rules [24] which are if –then rules that its conditions and decisions both consists of fuzzy variables that belongs to some fuzzy sets with some degree of membership.

IF x is A, THEN y is B

Where (x is A) and (y is B) are two fuzzy propositions; x and y are fuzzy variables defined over universes of discourse U and V respectively; and A and B are fuzzy sets defined by their fuzzy membership functions $\mu_A(u): U \rightarrow [0,1]$ and $\mu_B(v): V \rightarrow [0,1]$

A generalized form of the fuzzy rule is the following:

IF x_1 is A_1 AND x_2 is A_2 . . . AND x_k is A_k, THEN y is B,

Where x_1, x_2. . . x_k, y are fuzzy variables (attributes) over different universes of discourse Ux_1, Ux_2, . . . ,Ux_k, Uy and A_1, A_2,. . ., A_k, B are their possible fuzzy values over the same universes.

A set of fuzzy rules has the following form:

Rule 1: IF x_1 is A_1, 1 AND x_2 is A_2, 1 . . . AND x_k is A_k, 1, THEN y is B_1,

ELSE

Rule 2: IF x_1 is A_1,2 AND x_2 is A_2,2 . . . AND x_k is A_k, 2, THEN y is B_2,

ELSE

. . .

Rule n: IF x_1 is A_1, n AND x_2 is A_2, n . . . AND x_k is A_k, n, THEN y is B_n

Fuzzy inference methods and defuzzification

Fuzzy inference [24] is an inference method that uses fuzzy implication relations, fuzzy composition operators, and an operator to link the fuzzy rules. The inference process results in inferring new facts based on the fuzzy rules and the input information supplied.

Different reasoning strategies over fuzzy rules are possible. Most of them use the generalized modus ponens rule. The generalized modus ponens inference law applied over a simple fuzzy rule can be expressed as follows: (IF x is A, THEN y is B) and (x is A'), then (y is B') should be inferred. The compositional rule of inference is one way to implement the generalized modus ponens law:

$$B' = A' \circ A \rightarrow B = A' \circ R_{ab} \qquad (2)$$

Where:

- $\circ$ Is a compositional operator.
- R_{ab} is a fuzzy relational matrix representing the implication relation between the fuzzy concepts A and B.

A fuzzy inference method combines the results Bi' for the output fuzzy variable y inferred by all the fuzzy rules for a given set of input facts. In a fuzzy production system, which performs cycles of inference, all the fuzzy rules are fired at every cycle and they all contribute to the final result. Some of the main else-links between fuzzy rules are:

• OR-link: The results obtained by the different rules are "OR-ed", so the more that is inferred by any of the rules, the higher the resulting degree of the membership function for B'. MAX operation is applied to achieve this operation.

• AND-link: The final result is obtained after a MIN operation over the corresponding values of the inferred by all the rules or fuzzy membership functions.

Defuzzification

Defuzzification is the process of calculating a single-output numerical value for a fuzzy output variable on the basis of the inferred resulting membership function for this variable. Two methods for defuzzification are widely used:

1. The center-of-gravity method (COG). This method finds the geometrical centre y' in the universe V of an output variable y, which center "balances" the inferred membership function B' as a fuzzy value for y. The following formula is used:

$$Y' = \frac{\sum \mu_{B'}(u) * u}{\sum \mu_{B'}(u)} \qquad (3)$$

2. The mean-of-maxima method (MOM). This method finds the value y' for the output variable y which has maximum membership degree according to the fuzzy membership function B'. If there is more than one value which has maximum degree, then the mean of them is taken.

3.2 Self Organized Feature Map

The Self Organized Feature Map (SOFM) [2] [21] is an unsupervised neural network that is capable of learning its weights from its input vector without supplying it with the corresponding output vector. SOFM is called self organizing or self adoption because they are able to decide what features it will use to group the input data. SOFM is usually, a two-layered network where the neurons in the output layer are organized into either a one or two-dimensional lattice structure (Bose and Liang, 1996). The SOM solves difficult high-dimensional and nonlinear problems such as feature extraction and classification of images and acoustic patterns, adaptive control of robots, and equalization, demodulation, and error-tolerant transmission of signals in telecommunications [21] and in this research it is used to find the membership function for fuzzy variables subsets. Figure 2 represents a simple structure for the SOFM where the dimension d is the number of the input neurons in the input layer for the following input data vector $x_n=[x_{n1}\ x_{n2}\ \dots\ x_{nd}\ ]^T$ and The synaptic weight vector at neuron j in the output layer is denoted by $wj = [w_{j1}\ w_{j2}\ \dots\ w_{jd}]^T$, j = 1,2,. . . ,J, where J is the total number of neurons in the output layer and w_{jk}, k = 1,2,. . . ,d, is the connecting weight from the j^{th} neuron in the output layer to the k^{th} neuron in the input layer.

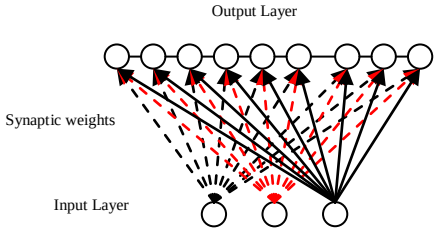

Figure 2: a simple structure for the SOFM

In the **learning phase**, the first step is to find the best matching neuron in the output layer that is the closest to the input vector from the following equation:

$$q(x_n) = \min_{\forall j} \left\| x_n - w_j \right\| \qquad (4)$$

Where,

$q(x_n)$ is the index of the winning neuron in the output layer,

x_n is the input vector,

w_j is the weight vector between the input vector and the output neuron j,

$\| \ \|$ is a distance measure (usually the Euclidean norm).

The next step is to update the weight vectors associated with the winning neuron q (x_n). The learning rule for neuron $j \in N_q$ where N_q is the chosen neighborhood of winning neuron q for input vector x_n, is given by

$$w_j[t+1] = w_j[t] + \eta_{qj}[t](x_n[t] - w_j[t]) \qquad (5)$$

Where

$$\eta_{qj}[t] = \begin{cases} \mu[t] & j \in N_q \\ 0 & j \notin N_q \end{cases} \qquad (6)$$

Here, μ[t] is the learning rate, 0 < μ [t] < 1, at time index t.

In the **retrieving phase**, when x_n is the input vector, only the winning neuron, after convergence, will have positive response.

3.3 Parallel Genetic Algorithms

Parallel genetic algorithms (PGAs) [23] [26] are parallel algorithms. Like the mechanism of sequential genetic algorithms (GAs) [4] [19] [22] [27], they are based on the natural evolutionary principle. PGAs have often been used for applications on massively parallel machines [18], transporters [20], and also on distributed systems [20]. However, the most important advantage of PGAs is that in many cases they provide better performance than single population-based algorithms, even when the parallelism *is simulated on conventional machines. The reason is that multiple populations permit speciation, a process by which different populations evolve in different directions (i.e. toward different optima) [5]. For this reason Parallel GAs are not only an

extension of the traditional GA sequential model, but they represent a new class of algorithms in that they search the space of solutions differently. In the PGA process better individuals survive and reproduce themselves more often than the worse ones. To speed up the processing of generations of populations, the population is split into several sub-populations and run the GA algorithm on each subpopulation in parallel way. After a number of iterations it calls the migrate process that enforce most fit individuals to travel from one sub population to another so as to make these sub populations much stronger. This migration process is enforced more than once during the PGA process (i.e. every particular number of iterations PGA calls the migrate process until end of iterations). The procedure of the PGA is illustrated in the following pseudo code.

Step1 initialize a set of sub population of solutions randomly.
Step2 for a number of iterations do the following steps
Step3 the selection reproduction process (select the fittest individuals from the generation according to some fitness function.
Step4 the cross over process (selects randomly two individuals and uses them as parents then randomly selects a crossover point inside the individuals , divide the two individuals and cross them together).
Step5 the mutation process (according to some mutation probability, pick one gene of any individual randomly and change its value from 1to 0 or 0 to 1 if we suppose that the individual is 1's and 0's).
Step6 if a predefined number of iteration passed call the migrate process
Step7 check the end of iterations if true go to step 7 else go to step 3
Step8 take the final population as the best solution.

In the PGA, The crossover operator, which recombines the bits (genes) of each two selected strings (chromosomes), has two kinds (the traditional and uniform crossover). Figure 3 shows the cross over process. The uniform cross over allows the system to create new rules with whatever possible antecedents. Traditional cross over needs some kind of antecedents reordering. The mutation also has two types (soft and hard mutation). Soft mutation changes the fuzzy set while the hard mutation changes the variable of the fact. Figure 4 shows the mutation process [11].

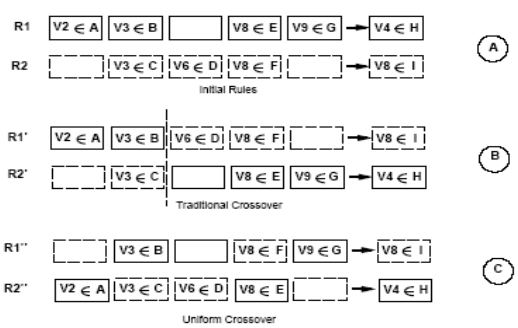

Figure 3 [11]: Crossover process

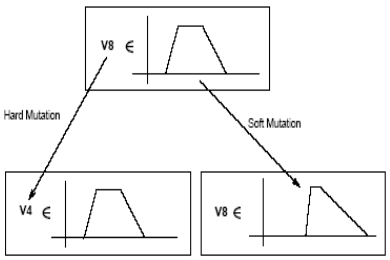

Figure 4 [11]: Mutation process

4. Self Organized Feature Map and Parallel Genetic Algorithms in Designing Fuzzy Rules

The fuzzy rules are simple if-then rules but with fuzzy variables. This research tries to build these fuzzy rules in two phases. The first phase is to generate the membership function for the subsets of the fuzzy variables. The second phase is to design the fuzzy rule using a set of training data and check for its applicability on the test data. Self Organized Feature Map and Parallel Genetic Algorithms were used for generating the membership functions [2] and designing the fuzzy rules respectively. The framework of the hybrid model that outlines the main modules is represented in figure 5. The data used in the training process as well as the features data collected from experts are used as inputs to the Generating fuzzy membership function for features subsets process which outputs a data file that contains the values of the fuzzy variables and their corresponding membership degrees in the subsets of these variables. These fuzzy membership degrees and the training data again are used as inputs to the generating fuzzy rules process which in turn outputs the corresponding fuzzy rules set after testing them by the test data records. The main components are:

Generating fuzzy membership function for features subsets: This module uses the SOFM capabilities of unsupervised learning and clustering to generate the membership functions of the features subsets.
Generating fuzzy rules: This module uses the PGA search mechanisms to build and search for the best fuzzy rules using its fitness function and then tests the resulting rules against the prepared testing data. Figure 10 shows a flow chart for the process.

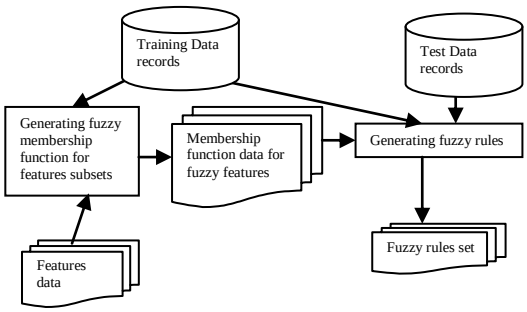

Figure 5: the framework of the hybrid model between the SOM & PGA to design fuzzy rules

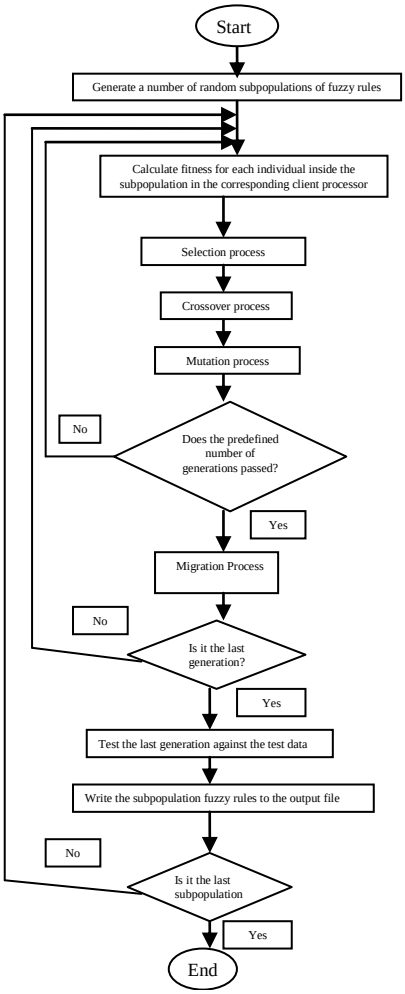

Figure 10: Generating fuzzy rules process

4.1 Generating Membership Functions Using Self Organized Feature Map

The process of generating membership functions used to be in two phases. The first phase generates the proper clusters of the feature data. Then, the fuzzy membership function is generated according these clusters. But it is possible to generate the membership functions in one phase by combining the variable labels with the variable values in the input layer of the SOFM [2]. As mentioned the SOFM learns from its input vector so the input vector will be X_n =(v , S_1,S_2,…S_d) where v is the value of the feature which the membership function of its fuzzy subsets wanted to be found and S_1,S_2…S_d are the subsets the feature will be divide to. These input vectors will be the training data which can be gathered by asking an expert some questions about the features like do you think that the value v of feature f belongs to s_1 or s_2 or …s_d? .The SOFM then goes through the learning phase and update its weights according to the learning procedure mentioned above. After convergence the weights of the SOFM will be the values of the variable and its corresponding membership degrees of the labels in the input layer. This technique is illustrated before in a research for Generating fuzzy membership function with self-organizing feature map by Chih-Chung Yang, N.K. Bose.

Example 1 [2] is a graphical example for illustrating the technique. Suppose that the fuzzy subsets for a variable like height would be ‘short’ and ‘tall’ and this section tries to find the membership function for each subset. The dataset could be collected by asking a question ‘‘Do you think a person with height 6 feet is tall or short?’’ After dataset is collected, the labeling information maybe represented by 2-D unit vectors [1 0]^T and [0 1]^T for fuzzy variable ‘short’ and ‘tall’, respectively. In the training phase of SOFM, the input feature height was combined with the labeling information to form a 3-D vector, which would be the input training sample for the SOFM. Suppose there are five neurons in the output layer as in figure 2 and the associated weights after training process are listed in Table 1. The fuzzy membership functions for the fuzzy variables tall and short are illustrated in figure 3. From this example the feature value height=5 has a degree of membership 0 in the subset tall and a degree = 1 in the subset short.

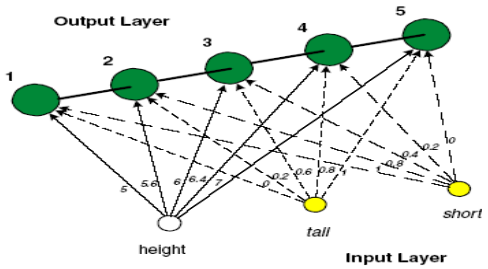

Figure 7 [2]: Fuzzy membership function for fuzzy subsets tall and short

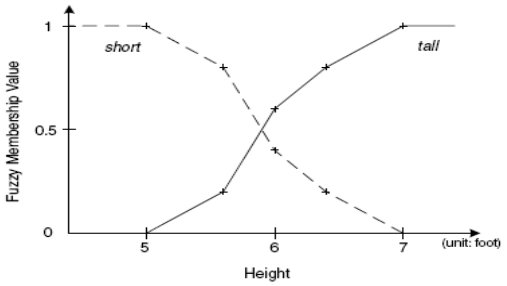

Figure 6 [2] SOFM after training process

Table 1: [2] Weights of SOFM after training process				
Neuron index	Associated weights	Feature height	Fuzzy variable Tall	Short
1	[5 0 1]^T	5	0	1
2	[5.6 0.2 0.7]^T	5.6	0.2	0.7
3	[6 0.6 0.4]^T	6	0.6	0.4
4	[6.4 0.8 0.2]^T	6.4	0.8	0.2
5	[7 1 0]^T	7	1	0

4.2 Generating Fuzzy rules using Parallel Genetic Algorithms

The process of generating fuzzy rules is responsible for designing the set of fuzzy rules of the knowledge based system. This process is conducted by applying the parallel processing power of PGA in randomly generating sub

populations of fuzzy rules at first then using some fitness function to choose the most fit performers from the sub populations and during the generations the populations becomes much more stronger. At the last generation the population itself becomes the set of fuzzy rules needed for the fuzzy inference engine of the knowledge based system. The PGA process in generating fuzzy rules and its fitness function is illustrated in the following sections.

The Parallel Genetic Algorithm Process

Since PGA which is a distributed evolutionary computing algorithm, shows high ability to find an admissible solution in a large search spaces. This property is used in the proposed hybrid model to find the best fuzzy rules from a random population of fuzzy rules according to a fitness function . The last population of the PGA will be the set of rules used later to classify new patterns using some inference engine. The individuals of this population will be the best fuzzy rules found. Each individual is composed of a number of genes corresponding to the features of the fuzzy system and the class depending on them. Figure 8 shows an individual (fuzzy rule) of a system that has 5 features and a depending class.

$f_1 \in s_2$	$f_2 \in s_3$	$f_3 \in s_2$		$f_5 \in s_1$	$c_1 \in s_1$

Figure 8: the PGA individual

The genes 1 to 5 are the features or independent variables of the system the 6th gene is the depending class. This individual is a fuzzy rule so both of its conditions and consequences are fuzzy propositions. This fuzzy rule assumes that

If $(f_1 \in s_2 \& f_2 \in s_3 \& f_3 \in s_2 \& f_5 \in s_1)$ then $c_1 \in s_1$

Note that f_1, f_2, f_3, f_5 are the fuzzy features and s_2, s_3, s_2, s_1 are the corresponding fuzzy subsets that the features belong to. Also this rule does not depend on f_4 and the resulting class is c_1 belonging to the fuzzy subset s_1.

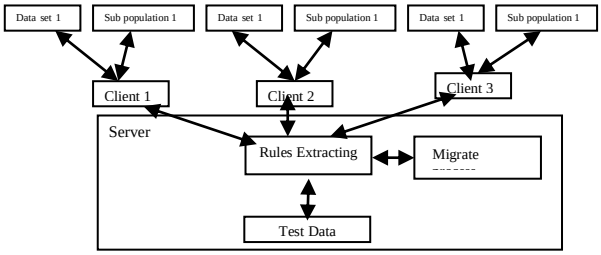

Figure 9: framework for the parallel distributed implementation of the rule generation process

The framework of the parallel distributed implementation of the generating fuzzy rules process is explained in figure 9. The training subsets along with the subpopulations are spread over the clients for the generations processing, after a number of iterations the subpopulations returns to the server for the migration process (that forces the best individuals to travel from sub population to an other). When all the generations finish, the rule extracting process tests the individuals against the test data to measure its accuracy level. Figure 10 shows the flow chart for the generating fuzzy rules process. This process starts by generating a number of random subpopulations to perform the PGA on them. Each sub population is a set of individuals as illustrated in figure 8. PGA process starts by assigning each client a subpopulation and a sub set of the training data and calling the calculate fitness for each individual process that use the fitness function to calculate how fit each individual is. Then the selection process selects the fit individuals and copies them in the next generation. The process then applies the crossover process, which yields two offsprings in the next generation from two different individuals (parents) in the current generation according to some crossover probability, and the mutation process, which yields a new individual in the next generation by simply changing a single gene in a random chromosome (individual) also according to some mutation probability. After a number of iterations the PGA returns to the server with all the sub populations and calls the migration process that forces the fit individuals to travel from one sub population to the other to make them stronger. The PGA checks for the end of iterations, if false it gives the subpopulations back to the clients for more iterations (repeats the steps again) else it picks up the subpopulations and extract the individuals without repetition and tests them against the test data.

Finding the fitness function

The fitness function determination is the most important step in the PGA or the GA because it is used to measure how fit each individual is and the selection process uses it to decide which individuals go from the current generation to the next. The confidence of classification rules is considered a very good measure for the strength of these rules. But some of the rules with high confidence levels could have a negative correlation between their conditional attributes (P) and the classes attribute (D). So it is preferred to take the fit rules as a conjunction between the rules with the highest confidence level and those that are positively correlated. The generating fuzzy rules process tends to select the rules with the highest confidence level (which exceeds a predefined threshold) and those with correlation coefficient between the conditional attributes and the classes attribute more than 1. Because the population is treated as a rule set used finally for the inference , all the rules (individuals) of that population will be fired for each data record. The Mumdani style max-min method is used to match the attributes with the instances in the training dataset. The membership degree is calculated by comparing the rules (containing the features and their subsets) with the data record. The minimum of the membership degrees is taken as the membership degree of that rule because the features inside the rules are connected by an AND-link. As the rules are connected by an OR-link inside the population, the maximum between all the membership degrees of the rules is taken as the rule representing that data record. The output of that rule is compared to the target output of the data record if they match then $P \wedge D$ matches else then P matches. Counting these matches of $P \wedge D$ & P helps in calculating the support equation that is used to calculate the confidence of that rule.

Fit individuals should have high confidence that exceeds a certain predefined threshold.

$$Confidence = \frac{\sup(P \wedge D)}{\sup(P)} \quad (7)$$

$$\sup(P \wedge D) = \frac{|P \wedge D|}{|N|} \quad (8)$$

$$\sup(P) = \frac{|P|}{|N|} \qquad (9)$$

Where P is the conditional attributes in rule R and D is the decision attribute.

$\sup(P \wedge D)$ Is the no of data records containing both P & D / total no of data records.

sup (P) Is the no of data records containing P / total no of data records.

The confidence rate is not a measure of the implication between P and D; however it is the conditional probability of D given P. So how to make sure that P implies D?

The correlation Coefficient is a measure for the certainty of the implication between P and D.

$$corr(P,D) = \frac{pr(P,D)}{pr(P) * pr(D)} \qquad (10)$$

$\text{corr}(P,D)$ <1 ==> A and B are negatively correlated.

$\text{corr}(P,D)$ >1 ==> A and B are positively correlated.

$\text{corr}(P,D)$ =1 ==> A and B are independent.

As the rule's conditional attributes should be positively correlated, the rules with high confidence values and correlation more than 1 is selected to be the most fit rules.

5. Experimental Results

The proposed hybrid model is composed of two main sub models. The first sub model is the Generating fuzzy membership functions for features subsets which is responsible for generating the degree of membership of the values of the variables in their corresponding subsets. This process uses the SOFM, which uses its unsupervised learning and clustering ability to learn the weights of the neural net from the input training data vectors, to obtain the variables values and their corresponding membership degrees. Using these values an analogue membership function for each subset of the variables can be drawn. This technique is implemented before in a previous work [2] and is used to prepare the fuzzy variables for the generating fuzzy rules process. The second sub model is the generating fuzzy rules module that applies the parallel distributed capabilities of the PGA to randomly build the fuzzy rule set in the initial sub populations and go through the iterations to choose the best rules representing the training data set according to the fitness function illustrated in section 4.2.2. The sub populations are composed of a set of individuals each individual represent a fuzzy rule. The final sub populations represent the final fuzzy rule set so they are just combined without repetition.

The proposed model is implemented in a parallel distributed client server environment where the process of generating the membership functions for the data sets variables is accomplished on the server and the process of generating fuzzy rules divided into sub processes some of them were implemented on the server and others on the clients. The main processes of the PGA like the initialization of sub populations , selection process, crossover and mutation processes were used in each client to produce new generations of rules but to keep these sub populations tied together the migration process is implemented on the server to make the best fit rules(individuals) travel from one sub population to the other .

After the whole generating process is finished the rule extractor process on the server extracts the final rules without repetition and tests them against the test data set for the accuracy measuring. The data sets used in this research to test the model are taken from the UCI machine learning repository and their properties are illustrated in table 2. The data set records are divided in two equal parts (one for the training data and one for the test data).

Table 2 : Description of the data sets properties

Name of the data set	No of attributes	No of continuous attributes	No of categorical attributes	No of data records	No of classes
weather	4	2	2	14	2
liver	6	6	0	345	2

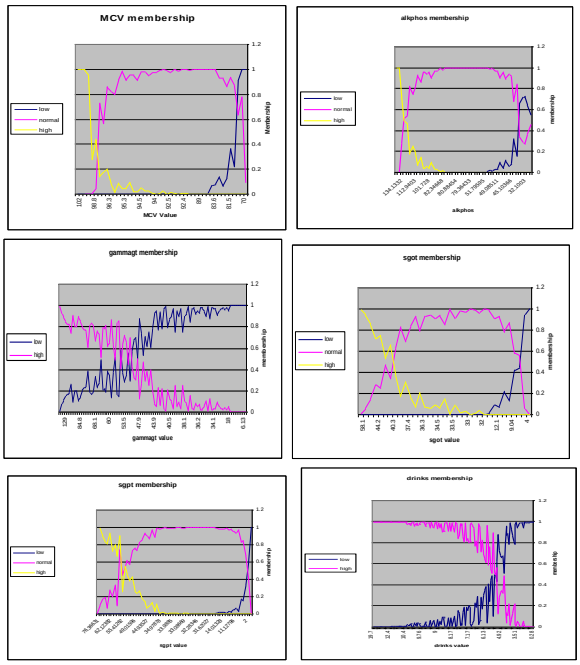

Figure 11: Membership functions of the liver data features from the SOM process

In experiments, the SOFM is trained with 3 or 4 input neurons (one for the feature value and the rest for the subsets) depending on the number of subsets of the features and 15 output neurons that produced 225 of features values and their corresponding membership degrees. Theses values were used to draw an analogue function for each feature. Figure 11 shows the membership functions of the 6 conditional features of the liver data set. After a number of trials the best parameters (No of generations, No of sub populations, Size of initial subpopulations, crossover rate, mutation rate and threshold) for the PGA process that help us in maximizing the strength (confidence level) of the fuzzy classification rules resulting in the final sub populations could be found as follows. Using a random initial fuzzy rules as the initial 10 sub populations with a size of 25 individual per sub population for the PGA in the process of generating the fuzzy rule set depending on 10 iterations to find the best generation in each sub population with 60% cross over rate, 0.01 a mutation rate and a threshold rate of 40 the frame work could

reach better fuzzy rule set accuracy than techniques used before in the generation of fuzzy rules. The list of these parameters is illustrated in table 3.

Table 3: Parameters values for PGA	
Parameter	value
No of generations	10
No of sub populations	25
Size of initial sub population	10
Crossover Rate	60%
Mutation Rate	0.01
threshold	40

The comparison between the proposed model and other techniques is listed in table 4. The table shows the accuracy levels of the rule sets generated by C4.5, Naïve Bays and the proposed model (SOFM + PGA) on two different data sets (weather and liver data sets). Figure 12 shows the same comparison in graphical mode.

Table 4 comparison between the proposed model and other techniques found in the field of generating fuzzy rules

	c4.5	naïve Bays	SOM +PGA
weather	68	64	*71.4*
liver	49	51	*60.7*

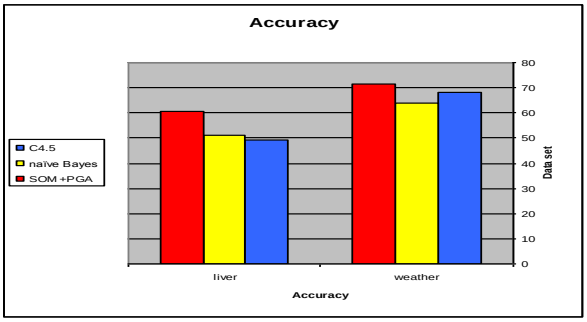

Figure 12: the accuracy of the rule set of the proposed model and some other rule generator algorithms

These comparisons show that the proposed hybrid model gave better accuracy level than the previous ones. The results indicate an average accuracy of around 66.05%, with accuracies above 71.4% even for quite small training sets. This compares favorably with previous systems for classifying documents, whose average accuracy is 58%.

6. Conclusions

Fuzzy Rules are a very good way to handle uncertainty in real world but designing them is a very challenging task. They depend on fuzzy propositions which in turn use fuzzy variables instead of regular ones. Two main problems appear in designing good, simple and accurate fuzzy rules. First finding the membership functions for the fuzzy subsets of the variables used in the fuzzy propositions, second arranging these propositions in if-then rules structure that efficiently represent the training data.

This research shows that the ability of SOFM to learn and cluster its inputs can help in clustering fuzzy variable into its corresponding fuzzy subsets and finding the representing fuzzy membership functions of these subsets and this could be a good solution for the first problem.

For the designing fuzzy rules problem, PGA as a good evolutionary searching mechanism that use more than one initial sub population to help finding the optimal fuzzy rule set after running the PGA on these sub populations. Each subpopulation consists of a set of randomly generated fuzzy rules as individuals and according to the fitness function the PGA chooses the best performers from all the subpopulations without repetition as the final fuzzy rule set used later for inference. These resulting fuzzy rules are further tested by means of testing data set to make sure that the accuracy meets the effective levels of performance. The experiment results are shown and they proved that the proposed hybrid model is much more efficient than previous techniques.

7. References

[1] A. A. Freitas, 2002, "Data Mining and Knowledge Discovery with Evolutionary Algorithms", Springer,Berlin.

[2] Chih-Chung Yang, N.K. Bose, 2006, "Generating fuzzy membership function with self-organizing feature map", Letters Volume, 1, Pages 356–365.

[3] Cordon O., F. A. C. Gomide, F. Herrera, F. Hoffmann and L. Magdalena, 2004, "Ten Years of Genetic Fuzzy Systems: Current Framework and New Trends", Fuzzy Sets and Systems, Pages 5-31.

[4] D. E. Goldberg, 1989, "Genetic algorithms in search, optimization, and machine learning", Addison-Wesley, 412.

[5] David E. Goldberg, 1989, "Sizing populations for serial and parallel genetic algorithms", Proc. of the Third International Conference on Genetic Algorithms, J. D. Schaffer, Ed., San Mateo, CA.

[6] D. L. A. Araujo, H. S. Lopes, and A. A. Freitas,2000, "Rule discovery with a parallel genetic algorithm ", Proc. of GECCO Workshop on Data Mining with Evolutionary Computation, pp. 89-92.

[7] Fevrier Valdez, Patricia Melin, Herman Parra, 2011, "Parallel genetic algorithms for optimization of Modular Neural Networks in pattern recognition", IJCNN ,pp.314-319.

[8] Fevrier Valdez, Patricia Melin, Oscar Castillo, 2011, "An improved evolutionary method with fuzzy logic for combining Particle Swarm Optimization and Genetic Algorithms". Appl. Soft Comput., Vol.11(2), pp.2625-2632.

[9] Fevrier Valdez, Patricia Melin, Oscar Castillo, 2010, "Evolutionary method combining Particle Swarm Optimisation and Genetic Algorithms using fuzzy logic for parameter adaptation and aggregation: the case neural network optimization for face recognition", IJAISC, Vol.2(1/2), pp.77-102.

[10] Ishibuchi H., T. Nakashima, and T. Murata, 1995, "A Fuzzy Classifier System that Generates Fuzzy If–Then Rules for Pattern Classification Problems", Proc. of 2nd

IEEE Int. Conf. Evolutionary Computation, Perth, Australia, pp. 759–764.

[11] Ishibuchi H., K. Nozaki,and H. Tanaka, 1996, "Adaptive Fuzzy Rule-Based Classification Systems", IEEE Trans. on Fuzzy Systems, vol. 4, no. 3, pp. 238-250.

[12] Ishibuchi H., T. Nakashima and T. Murata, 1999, "Performance Evaluation of Fuzzy Classifier Systems for Multi-Dimensional Pattern Classification Problems", IEEE Trans. Syst., Man, Cybern, Part B, vol. 29, pp. 601-618.

[13] J. H. Holland, 1975, "Adaptation in natural and artificial systems", University of Michigan Press, Ann Arbor, MI.

[14] Juan R. Velasco and Luis Magdalena, 1995, "Genetic Algorithms in Fuzzy Control Systems", In J. Periaux and G. Winter, editors, Genetic Algorithms in Engineering and Computer Science. John Wiley & Sons Ltd, pages 141-165, ISBN 047195859 X.

[15] Lotfi A. Zadeh, 2002, "From computing with numbers to computing with words —from manipulation of measurements to manipulation of perceptions", in International Journal of Applied Math and Computer Science, pp. 307–324, vol. 12, no. 3.

[16] Lotfi A. Zadeh, 1965, "Fuzzy sets and systems". In: Fox J, editor. System Theory. Brooklyn, NY: Polytechnic Press, pp. 29–39.

[17] Reiko Tanese, 1989,"Distributed Genetic Algorithms for Function Optimization", Ph.D. thesis, University of Michigan, Computer Science and Engineering.

[18] Reiko Tanese, 1987, "Parallel genetic algorithms for a hypercube", Proc. of the Second International Conference on Genetic Algorithms.

[19] Saroj, Nishant Prabhat, 2011," A Genetic-Fuzzy Algorithm to Discover Fuzzy Classification Rules for Mixed Attributes Datasets", International Journal of Computer Applications, Vol 34– No.5.

[20] T. C. Fogarty and R. Huang, 1991, "Implementing the genetic algorithm on transputer based parallel processing systems", in Parallel Problem Solving from Nature, Springer Verlag, Berlin, Germany, pp. 145–149.

[21] T. Kohonen, 2001, "Self-Organizing Maps", Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York, ISBN 3-540-67921-9, ISSN 0720-678X 1995, 1997.

[22] M. Vose, 1999, "The Simple Genetic Algorithm Foundation and Theory", MIT Press, 251.

[23] Mariusz Nowostawski, Riccardo Poli, 1999, "Parallel Genetic Algorithms Taxonomy", Proceedings of Third International Conference on Knowledge-based Intelligent Information Engineering Systems KES'99 Adelaide, South Australia, 31 August - 1 September.

[24] Nikola K. Kasabov, 1996, "Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering", the MIT Press, Cambridge, MA, ISBN 0-262-11212-4.

[25] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, 2004, "Ten years of genetic fuzzy systems: Current framework and new trends", Fuzzy Sets and Systems, pp. 5-31.

[26] Z. Konfršt, 2004, "Parallel Genetic Algorithms: advances, computing trends, application and perspectives", 18th IPDPS 2004, IEEE CS, Santa Fe, New Mexico, pp.162.

[27] Z. Michalewicz, 1992, "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlang, 252.

[28] Yuan Yufei and Zhuang Huijun , 1996 ,"A Genetic Algorithm for Generating Fuzzy Classification Rules", ELSEVIER Fuzzy Sets, vol. 84, pp. 1-19.

[29] Yusuke Nojima, Hisao Ishibuchi, Isao Kuwajima, 2009, "Parallel distributed genetic fuzzy rule selection", Soft Computing, Vol. 13(5), pp.511-519.