Learning Fuzzy Network Using Sequence Bound Global Particle Swarm Optimizer

Satchidananda Dehuri, Fakir Mohan University, India

Sung-Bae Cho, Yonsei University, Korea

ABSTRACT

This paper proposes an algorithm for classification by learning fuzzy network with a sequence bound global particle swarm optimizer. The aim of this work can be achieved in two folded. Fold one provides an explicit mapping of an input features from original domain to fuzzy domain with a multiple fuzzy sets and the second fold discusses the novel sequence bound global particle swarm optimizer for evolution of optimal set of connection weights between hidden layer and output layer of the fuzzy network. The novel sequence bound global particle swarm optimizer can solve the problem of premature convergence when learning the fuzzy network plagued with many local optimal solutions. Unlike multi-layer perceptron with many hidden layers it has only single hidden layer. The output layer of this network contains one neuron. This network advocates a simple and understandable architecture for classification. The experimental studies show that the classification accuracy of the proposed algorithm is promising and superior to other alternatives such as multi-layer perceptron and radial basis function network.

Keywords: *Classification, Fuzzy Network, Multi-Layer Perceptron, Particle Swarm Optimization, Radial Basis Function Network*

INTRODUCTION

In the past few decades, information technology and the World Wide Web (WWW) have created stacks of innovations in the area of marketing style of companies. More businesses and companies are collecting highly informative and valuable data in a large scale. The huge amount of data can be a gold mine for business management and marketing. It is therefore increasingly important to analyze the data. However, timely and accurately processing tremendous volume of data with traditional methods (Michie et al., 1994) is a difficult task. For example, using multi-layer perceptron (MLP) in data mining is not likely produce any useful results (Edelstein, 1996), because it does not have a clean interpretation of the model and its longer training time can make frustration. The ability to analyze and utilize massive data lags far behind the capability of gathering and storing it. This gives rise to new challenges for businesses and researchers in the extraction of useful information.

DOI: 10.4018/ijfsa.2012010104

Data mining-a core step of knowledge discovery in databases (Piatetsky-shapiro et al., 1996; Han & Kamber, 2001), is defined as a process of employing one or more computer learning techniques to automatically analyze and extract knowledge from vast amount of data contained within a database. The purpose of data mining is to identify trends and patterns in data. Classification (Duda et al., 2001) is one of the widely used techniques of data mining. In addition to classification, there are many important tasks such as association rule mining (Agrawal, 1993), clustering (Jains & Dubes, 1988), regression analysis (Chen, 2011), summarization, etc. in the area of data mining. However, classification is a fundamental activity in pattern recognition (Theodoridis & Koutroumbas, 2006; Bishop, 2006), data mining and so forth. Given predetermined disjoint target classes $\{C_1, C_2,...,C_n\}$, a set of input features $\{F_1, F_2, .., F_m\}$ and a set of training data T with each instance taking the form $<a_1, a_2, ..., a_m>$, where a_i $<i=1,2,..., m>$ is in the domain of attribute F_i, $i=1,2,\ldots,m$ and associated with a unique target class label the task is to build a model that can be used to predict the target category for new unseen data given its input attribute values.

There are many classifiers like naïve Bayes (Robert, 2001; Winkler, 2003; Gelman, et al., 1995), linear discriminant (McLachlan, 2004), k-nearest neighbour (Dasarathy, 1990; Herbrich, 2001), MLP (Bishop, 1995) and its variant, decision tree (Breiman et al., 1984; Quinlan, 1992), and many more are commonly available in the specialized literatures. We can use the existing techniques of pattern recognition for classification in the context of data mining but these algorithms were designed only for small dataset. Therefore, either we can prefer to design a new algorithm or we can reengineer the existing classification algorithms of pattern recognition such that a gamut of data can handle efficiently. Data mining does not compete with traditional methods. However, it offers better solutions in certain classes problems than traditional methods. Data mining methods and algorithms extract useful regularities from large data archives, either directly in the form of knowledge or indirectly as functions that allow predicting, classifying or representing regularities in the distribution of the data.

A neural network classifier like MLP is a parallel computing system of several interconnected processor nodes. The strength of MLP is the ability to construct non-linear boundary with high classification accuracy. However, the main weakness of this network lies in its architectural complexity and training speed. It requires many passes to build. This means that creating the most accurate models can be computationally expensive. Craven and Shavlik (1997) was described a neural network learning algorithm that can suitably address the issues of model comprehensiveness and training speed. However, it leaves several rooms for carrying out more research on noisy and uncertain data.

The paper by Bellman et al. (1996) was the starting point in the application of fuzzy set theory to classification. Since then, researchers have found several ways to apply this theory to generalize the existing classification methods, as well as to develop new algorithms. There are two main categories of fuzzy classifiers: fuzzy *if-then* rule-based and non *if-then* rule fuzzy classifiers. The second group may be divided into fuzzy k-nearest neighbours and generalized nearest prototype classifiers (GNPC). Several approaches (Nauck & Kruse, 1997; Uebele et al., 1995; Chakraborty & Pal, 2004; Abe & Thawonmas, 1997; Roubos et al., 2003) have been proposed for automatically generating fuzzy *if-then* rules and tuning parameters of membership functions for numerical data. These methods fall into three categories: neural-network-based methods with high learning abilities, genetic (evolution)-based methods with the Michigan and Pittsburg approaches, and clustering-based methods. There are several methods (e.g., Zogala, 2000; Ishibuchi et al., 2001; Zhou & Khotanzad, 2007) that combine the categories that have proved as effective in improving classification performance.

Recently, a new direction in the fuzzy classifier design field has emerged: a combination of multiple classifiers using fuzzy sets (Kuncheva, 2002), which may be included in non *if-then* fuzzy classifier category. In general, there are two types of the combination: classifier selection and classifier fusion. In the first approach each classifier is an *expert* in some local area of the feature space. In the second approach all classifiers are trained over the whole feature space. Thus, in this case, we have competition, rather than complementation, among the fuzzy classifiers. Various methods (Chen, 2005; Luo et al., 2007; Hong et al., 2007) have been proposed for fuzzy classifier design; however, in contrast to statistical and neural pattern classifiers, both theoretical and experimental studies concerning fuzzy classifiers do not deal with the analysis of the influence of the classifier complexity on the generalization error.

Considering the best attributes of neural network, fuzzy, and swarm theory in this paper, we present a sequence bound global particle swarm optimizer technique to obtain a scalable classifier with high accuracy. Particle swarm optimization (PSO) (Kennedy & Eberhart, 2001; Kennedy & Mendes, 2002) inspired by natural flocking and swarm behavior of birds and insects have gain popularity as an optimization tool. Although we can use other evolutionary techniques like genetic algorithms (GAs) (Goldberg, 1989; Michalewicz, 1994), compared to GAs, PSO algorithm (Liang et al., 2006; Parrot & Li, 2006) possesses many attractive properties such as memory and constructive cooperation between individuals. Hence, it has more chance to fly into the better solution areas more quickly and discover reasonable quality solution much faster.

In PSO (Liang et al., 2006; Parrot & Li, 2006) only few parameters are to be tuned. Unlike GA, the representations of the weights are difficult and the genetic operators have to be carefully selected or developed. Further, as there is no selection operator in PSO, each particle in an original population has a corresponding partner in a new population. From the point of the diversity of population, this property is better than GA, so it can avoid the premature convergence and stagnation in GA to some extent. The proposed technique contains a simple perceptron. The input data is converted with the fuzzy membership functions, which are then multiplied with the weights evolved by our novel sequence bound global particle swarm optimizer and fed to the perceptron. The sequence bound particle swarm optimizer is used for evolving a set of optimal weights for a particular architecture using the mean square error (MSE) criterion. The sequence bound global particle swarm optimizer is designed by limiting the velocity using a sequence bound method (Barrera & Coello, 2009) and the comprehensive learning particle swarm optimizer (Liang et al., 2006). The comprehensive learning particle swarm optimizer uses a novel learning strategy whereby all other particles historical best information is used to update a particle's velocity. This strategy enables the diversity of the swarm to be preserved to discourage premature convergence.

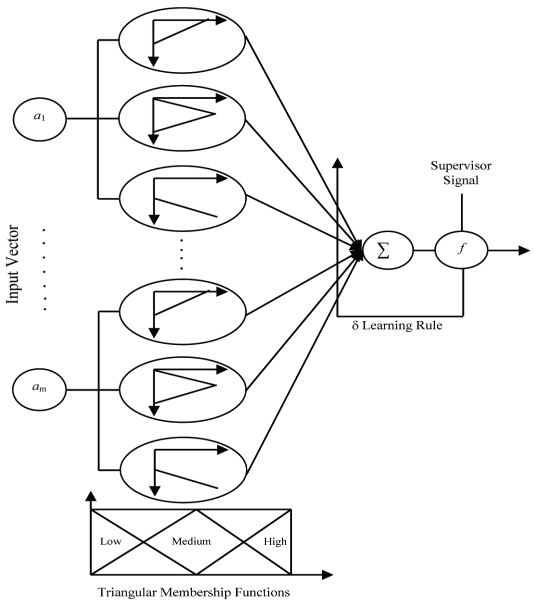
The rest of the paper is organized as follows. The preliminary concepts of fuzzy network architecture and particle swarm optimization is briefly discussed. We discuss the proposed sequence bound global particle swarm optimizer for fuzzy network classifier. We cover the results obtained during experimental studies. Finally, with a point of future research directions the article is concluded.

PRELIMINARY

Fuzzy Network Architecture

The fuzzy network (Watanabe et al., 1993) is basically a flat network with only one hidden layer and hence, the learning algorithm used in this network becomes easy to comprehend. The architecture of this network model uses a simple perceptron. The input vector is expanded into a set of fuzzy sets depending on the characteristics of the given data points. Each point of the input vector is mapped into the fuzzy sets. Therefore, the dimension of the given input vectors is mapped from lower to higher dimensions with fuzziness. Since the given input space is divided into fuzzy sub-spaces by the membership functions, the network has an abil-

Figure 1. Architecture of fuzzy network using three triangular membership functions



ity of non-linear mapping. The expanded input vectors effectively increase the dimensionality of the input vector and hence the hyper planes generated by the fuzzy network provide greater discrimination capability. Figure 1 shows a general architecture of a fuzzy network with three triangular membership functions.

In this architecture, each unit of the input layer is mapped to the triangular membership functions of the hidden layer. There are no connection weights in between input and hidden layer. The given input vector is just a simple mapping from crisp domain to fuzzy domain. Each input vector <a_1, a_2, …, a_m> is the input of these membership functions (small, medium, and high) and their corresponding outputs O_{ij} (i=1,2,…,m, and j=1,2,3) are the grades of the membership functions. These outputs are multiplied with weight vector and given to output layer neuron as an input, i.e., $O_{ij}.w_{ij}$, i=1,2,..,m and j=1,2,3. The output unit has a sigmoidal function f_{sig} given by: $f_{sig}(S) = \hat{y} = 1/(1+e^{-\lambda S})$, where S is the sum of the inputs of the output unit, $S = \sum_{i=1}^{m}\sum_{j=1}^{3} O_{ij}.w_{ij}$ and $\hat{y}$ is the output of the network. The connection weights are modified by the delta-learning rule. Let $w_{ijk}(l)$ denote the value of synaptic weight w_{ijk} of neuron k excited by element $O_{ijk}(t)$ of the ith input vector $\vec{x}_i$ at time step t. according to delta learning rule, the weight adjustment $\Delta w_{ijk}(t)$ applied to the synaptic weight w_{ijk} at time step t is denoted by: $\Delta w_{ijk}(t) = \eta.e_i(\vec{w}).O_{ijk}(t)$, where η is a positive constant that determines the rate of learning as we proceed from one step in the learning process to another and $e_i(\vec{w})$ is the error function. In other words, the delta learning rule may be stated as "The adjustment made to a synaptic weight of a neuron is proportional to the product of the error signal and the input signal of the synapse in question" (Haykin, 1999).

Consider a set of records $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$, each of which consists of a set of attributes $\vec{x}_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$. For any record $\vec{x}_i \in X$, x_{ij} denotes the value of the j^{th} attribute of i^{th} record. Each x_{ij} is mapped to the fuzzy sets by the pre-defined sets of fuzzy membership functions such as $f_m = \left\langle f_{low}, f_{med}, f_{high} \right\rangle$. As a result of this mapping each input vector transferred from crisp domain to fuzzy domain with higher dimension. Figure 2 illustrates an example of a simple mapping of a sample $\vec{x}_i$.

The number of fuzzy sets can be chosen as per the complexity of the problem. Increasing more number of fuzzy sets does not mean that the accuracy of the model will increases in contrast it degrades the overall performance of the system. In the context of data mining the discovered knowledge should not only accurate but also comprehensible to the user. Comprehensibility is important whenever discovered knowledge will be used for supporting a decision made by a human user. It is also fact that if the model is very complex than the comprehensibility decreases; as a result the user will not be able to interpret and validate it. Hence identifying optimal and the type of fuzzy sets is also an important research direction.

In order to use this architecture for classification, we modified this by introducing Gaussian membership functions and replace the delta-learning rule by sequence bound particle swarm optimization based learning. We will discuss the details procedure of the proposed method. A good survey of fuzzy neural networks can be found in Auckley and Hayashi (1994). Further, some of the related and recent work of fuzzy neural networks is provided in Wu et al. (2010), Park (2002), and Chen, Wu, and Wang (2009).

Particle Swarm Optimization

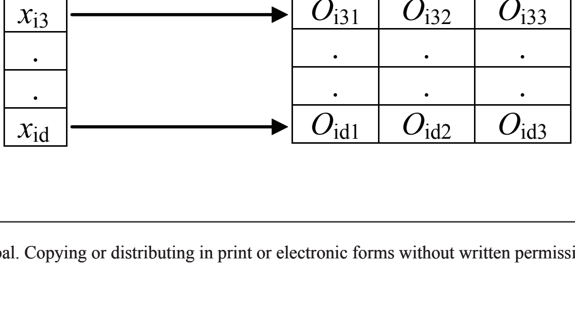
The particle swarm optimization algorithm is inspired by the metaphor of social interaction observed among insects or animals. The kind of social interaction modeled within a PSO is used to guide a population of individuals (called particles) moving towards the most promising area of the search space. In a PSO algorithm, each particle is a candidate solution equivalent to a point in a *d*-dimensional space, so the i^{th} particle can be represented as $x_i = (x_{i1}, x_{i2,} \ldots, x_{id})$. Each particle "flies" through the search space, depending on two important factors, pbest$_i$ = $p_i = (p_{i1}, p_{i2,} \ldots, p_{id})$, the best position found so far by the current particle and gbest = $p_g = (p_{g1}, p_{g2,} \ldots, p_{gd})$, the global best position identified from the entire population (or within a neighborhood). The rate of position change of the i^{th} particle is given by its velocity $v_i = (v_{i1}, v_{i2} \ldots v_{id})$. Equation (1) updates the velocity for each particle, whereas equation (2) updates each particle's position in the search space.

$$v_{id}(t) = v_{id}(t-1) + c_p.rand_1.(p_{id}(t-1) \\ -x_{id}(t-1)) + s_p.\text{rand}_2\ (p_{gd}(t-1) - x_{id}(t-1)) \tag{1}$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \tag{2}$$

where *t* denotes the t^{th} iteration; c_p and s_p are positive constants called the cognitive and social parameters respectively, and in general both the

Figure 2. Mapping of an instance from crisp domain to fuzzy domain



Algorithm 1. PSO_Algorithm

```
1) Initialize the swarm, P (t), of particles such that the position $\vec{x}_i(t)$
of each particle $p_i$ ∈ P(t) is random within the hyperspace, with t =0.
2) Evaluate the performance F of each particle, using its current position
$\vec{x}_i(t)$ .
3) Compare the performance of each individual to its best performance thus
    a. far: if f( $\vec{x}_i(t)$ ) < $pbest_i$ then,
    b. $pbest_i$ = f( $\vec{x}_i(t)$ ).
$\vec{x}_{pbest_i} = \vec{x}_i(t)$ .
4) Compare the performance of each individual to the global best particle:
    a. if f( $\vec{x}_i(t)$ ) < gbest then,
    b. gbest = f( $\vec{x}_i(t)$ ).
$\vec{x}_{gbest} = \vec{x}_i(t)$ .
5) Change the velocity vector for each particle using:

$\vec{v}_i(t) = w \times \vec{v}_i(t-1) + c_p \times rand_1 \times (\vec{x}_{pbest_i} - \vec{x}_i(t)) + s_p \times rand_2 \times (\vec{x}_{gbest} - \vec{x}_i(t))$ .
6) Move each particle to a new position:
    a. $\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$
    b. t = t+1
7) Go to step 2, and repeat until convergence.
```

value of c_p and s_p are equal to 2; $rand_1$ and $rand_2$ are random numbers uniformly distributed in the range (0,1).

A parameter called inertia weight w is introduced by Shi and Eberhart (1998, 1999) in PSO to balance the global and local search ability of the particle. Therefore, equations (1) and (2) are modified as follows.

$$v_{id}(t) = w.v_{id}(t-1) + c_p.rand_1.(p_{id}(t-1) \\ - x_{id}(t-1)) + s_p.\mathrm{rand}_2\ (p_{gd}(t-1) - x_{id}(t-1)) \tag{3}$$

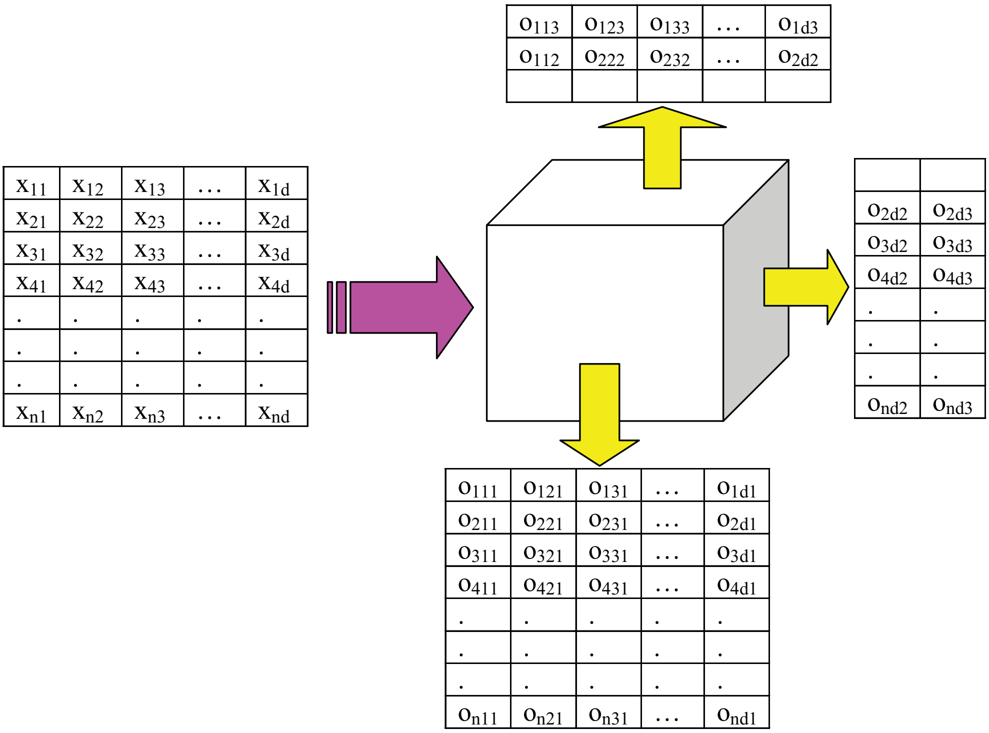
$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \tag{4}$$

A large inertia weight facilitates a global search while a small inertia weight facilitates local search. Shi et al. (1998, 1999) avoided that by linearly decreasing the inertia weight from a relatively large value to a small one through the course of the PSO run, the PSO tends to have more global search ability at the beginning of the run and have more local search ability near the end of the run. They suggested that an inertia weight starting from 0.9 linearly decreasing to 0.4 during a run be adopted to give the PSO a better performance. In some cases, an upper bound v_{max} and a lower bound v_{min} are used to clamp the velocities of the particles. A too small value of speed will cause the particles to get trapped in local optima; on the other hand, a too large value can cause the particle to oscillate around a position (Ozcan & Mohan, 1998).

The following algorithmic form of the global best version, *g*best, of PSO reflects the star neighborhood sociometry structure. The social knowledge used to drive the movement of particles includes the position of the best particle from the entire swarm. In addition, each particle uses its history of experiences in terms of its own best solution thus far.

The further away a particle is from the global best position and its own best solution thus far, the larger the change in velocity to move the particle back toward the best solutions.

Figure 3. A cubical view mapping pattern of X from crisp domain to fuzzy domain



o_{113}	o_{123}	o_{133}	...	o_{1d3}
o_{112}	o_{222}	o_{232}	...	o_{2d2}

x_{11}	x_{12}	x_{13}	...	x_{1d}
x_{21}	x_{22}	x_{23}	...	x_{2d}
x_{31}	x_{32}	x_{33}	...	x_{3d}
x_{41}	x_{42}	x_{43}	...	x_{4d}
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
x_{n1}	x_{n2}	x_{n3}	...	x_{nd}

o_{2d2}	o_{2d3}
o_{3d2}	o_{3d3}
o_{4d2}	o_{4d3}
.	.
.	.
.	.
o_{nd2}	o_{nd3}

o_{111}	o_{121}	o_{131}	...	o_{1d1}
o_{211}	o_{221}	o_{231}	...	o_{2d1}
o_{311}	o_{321}	o_{331}	...	o_{3d1}
o_{411}	o_{421}	o_{431}	...	o_{4d1}
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
o_{n11}	o_{n21}	o_{n31}	...	o_{nd1}

OUR METHOD

In contrast to simple fuzzy network (Watanabe et al., 1993), the major contributions of the proposed method is that it extends the application domain from feature extraction to classification of data in data mining domain by using the Gaussian membership functions and our novel sequence bound global particle swarm optimizer based learning. The proposed method is divided into two phases. In the first phase the given input vector is mapped from crisp to fuzzy domain. Figure 3 shows the schematic diagram of mapping the dataset *X* to fuzzy domain of higher dimension. In the second phase the sequence bound particle swarm optimizer is described and used for training the network by evolving an optimal set of weights. The two phases will continue until the proposed network is trained enough.

Let the database $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ represent a matrix of size $n \times d$, then $X = (x_{ij})_{n \times d}$, *i*=1, 2, 3, …, *n* and *j*=1, 2, 3, .., *d*. Each element of the matrix is mapped into the following three Gaussian membership functions.

$$O_{ij1}(x_{ij}, sm_j, th_j) = \exp(-\tfrac{1}{2}(x_{ij} - sm_j)^2 \,/\, (th_j)^2) \tag{5}$$

$$O_{ij2}(x_{ij}, me_j, th_j) = \exp(-\tfrac{1}{2}(x_{ij} - me_j)^2 \,/\, (th_j)^2) \tag{6}$$

$$O_{ij3}(x_{ij}, hg_j, th_j) = \exp(-\tfrac{1}{2}(x_{ij} - hg_j)^2 \,/\, (th_j)^2) \tag{7}$$

where x_{ij} is the j^{th} attribute value of the i^{th} record, sm_j is the low value, hg_j is the high value and me_j is the middle value of the j^{th} input feature and th_j is taken as $(hg_j - sm_j) / 3$. Each element of the input vectors is the input of these membership functions (low, medium, and high) and the outputs of the units of $\vec{O}$ are the grades of the membership functions. The inputs of the unit in the output layer are $O_{ijk} \times w_{ijk}$, where i=1,2,..,n, j=1,2,..,d, and k=1,2,3. The output unit has a sigmoid function f_{sig} given by

$$\hat{y}_i = f_{sig}(s) = {}^{1}\!\!\diagup_{(1+\exp(-\lambda s))},$$

where λ is constant and s is the sum of the inputs of the output unit and for a particular record the value of s can be written as:

$$s = \sum_{j=1}^{d} \sum_{k=1}^{3} O_{ijk} . w_{ijk} .$$

The error e_{i} obtained from the i^{th} vector can be estimated as $e_i(\vec{w}) = {}^{1}\!\!\diagup_{2}\,(t_i - y_i)^2$

$$\Rightarrow e_i(\vec{w}) = {}^{1}\!\!\diagup_{2}\,(t_i - f_{sig}(\sum_{j=1}^{d} \sum_{k=1}^{3} O_{ijk} \times w_{ijk}))^2 .$$

The mean squared error (MSE) is given by $E(\vec{w}) = {}^{1}\!\!\diagup_{n} \sum_{i=1}^{n} e_i(\vec{w})$. Using this criterion as the fitness function, sequence bound global particle swarm optimizer evolves the optimal set of weights for the proposed algorithm.

Sequence Bound Global Particle Swarm Optimizer

The sequence bound global particle swarm optimizer is constructed by limiting the velocity of the particle by sequence bound and updating the velocity of any one particle by considering all particles' personal bests.

Let us see how we can update the velocity by the novel strategy. In this learning strategy, we use the following updating equation:

$$v_{id}(t) = w.v_{id}(t-1) + \\ c_p.rand\ .(p_{f(p_{id})}(t-1) - x_{id}(t-1)) \qquad (8)$$

where $f(p_{id})$ defines which particles' personal bests the particle i should follow. $p_{f(p_{id})}$ can be the corresponding dimension of any particles' personal best including its own personal best, and the decision depends on probability τ referred as learning probability, which can take different values for different particles. For each dimension of particle i, we generate a uniform random number from [0, 1]. If this random number is larger than τ, the corresponding dimension will learn from its own personal best; otherwise it will learn from another particle's personal best. We employ the tournament selection of different neighborhood sizes when the particle's dimension learns from another particle's personal best as follows. The neighborhood size varies from large to small one (i.e., of 2) as the iteration increases.

1. Randomly choose two particles (e.g., the neighborhood size is 2) out of the population which excludes the particle whose velocity is updated.
2. Compare the fitness values of these two particles' personal bests and select the better one. As the problem is a minimization problem so we define the fitness value smaller the better.
3. Use the winner's personal best as the exemplar to learn from for that dimension. If all exemplars of a particle are its own personal best, we will randomly choose one dimension to learn from another particle's personal bests corresponding dimension.

Limiting the velocity of the particles is computed at each generation according to the corresponding element of a sequence of terms obtained from a geometric series. A geometric series V is a sum of algebraic terms and is given by the equation (9).

$$V = \sum_{i=0}^{\infty} lr^i = \tfrac{l}{1-r}, \qquad (9)$$

The geometric series V converges provided that |r| < 1 and *l* is a constant value. The ordered list of terms of a series with $i = 0$ to infinity is called a sequence. A known result is that, if the geometric series defined in equation (9) converges, then the elements of its sequence converge to zero.

We construct a limit for the velocities as follows: at the iteration t, the ith coordinate of the velocity of a particle is bounded by the term $r^t l_i$, where r is a constant value with $0 < r < 1$ in order to guarantee the convergence to zero of the bound, and $l_i = [x_{\max_i} - x_{\min_i}]$ is the longitude of the search space in the ith coordinate with $i = 1 \ldots n$. To compute the velocity we use equation (8).

After computing $v_{id}(t)$ and before computing the position, we limit the velocity of each particle with the term $r^t l_i$.

During each iteration we calculate the error for all the particles in the distributed environment. The stopping criterion may allow the particles to iterate till they converge to a single decision. However, in this process, the particle gets over fitted and leading to poor performance of the classifier. Therefore, we need to be careful about choosing the stopping criterion of the proposed method.

The following high-level pseudocode of the proposed method can give how the idea has been experimented and also validated.

EXPERIMENTAL STUDIES

In this section the performance of the proposed model is evaluated using the real world benchmark classification datasets. The most frequently used in the area of neural networks and neuro-fuzzy systems are Wisconsin Breast Cancer (WBC), PIMA, WINE and BUPA Liver Disorders. All these datasets are taken from the UCI machine learning repository (Blake & Merz, 1998).

Description of the Datasets

Let us briefly discuss the datasets, which we have taken for our experimental setup.

- *Wisconsin Breast Cancer Dataset*: The dataset consists of *d*=9 features made on each of the 699 clinical cases of class *c*=2. The two distinct categories correspond to two different classes such as *Benign* and *Malignant.* The problem is to classify each test point to its correct cases based on the nine test.
- *PIMA Indians Diabetes Dataset*: This dataset consists of *n* = 8 numerical medical attributes and *c*=2 classes (tested positive or negative for diabetes). There are *n*=768 instances. Further, data set related to the diagnosis of diabetes in an Indian population that lives near the city of Phoenix, Arizona.
- *BUPA Liver Disorders Dataset*: This dataset is related to the diagnosis of liver

Table 1. Descriptions of the datasets

Dataset	Number of Patterns	Number of Attributes	Number of Classes	Patterns in Class 1	Patterns in Class 2	Patterns in Class 3
WBC	699	10	2	458	241	
PIMA	768	8	2	500	268	
BUPA	345	6	2	145	200	
WINE	178	13	3	59	71	48

Table 2. Results obtained from the proposed method

Dataset	Hit Rate % in Training Set	Hit Rate % in Test Set
WBC	97.0858	95.5714
	97.9656	97.1346
Mean Value	**97.5257**	**96.353**
PIMA	81.0156	75.9376
	79.4532	75.1302
Mean Value	**80.2344**	**75.5309**
BUPA	75.3486	70.1745
	76.9368	68.1502
Mean Value	**76.1428**	**69.1476**
WINE	96.46065	98.3707
	93.31455	98.3145
Mean Value	**94.8876**	**98.3426**

disorders and created by *BUPA Medical Research, Ltd.* The dataset consists of d=5 attributes and c=2 classes. There are n=345 number of instances.

- *WINE Dataset*: This is a three-class problem. It has 178 instances and 13 attributes. The instances are distributed into three classes. Class 1 contains 59, class 2 contains 71 and class 3 contains 48 patterns.

Pseudo Code

```
1. Initialize weight vectors $w_i$ of the Swarm
2. Repeat (Generations)
3. For each weight vector (i.e., Particle)
4.    For each sample from the training vector
       4.1 Compute $e_i(\vec{w})$;
       4.2 SUM=SUM+ $e_i(\vec{w})$;
5.   End For
6. Compute the mean squared error $E_i(\vec{w})$;
7. End For
8. Compute velocities of the Swarm by equation (8);
9.  Limit the velocities;
10. Update the position of the Particle;
11. Until (Convergence occurs)
```

Table 3. Performance comparison of proposed method, RBF and MLP

Dataset	**MLP**		**RBF**		**Proposed Method**	
	Hit % in Training Set	**Hit % in Test Set**	**Hit % in Training Set**	**Hit % in Test Set**	**Hit % in Training Set**	**Hit % in Test Set**
WBC	97.0858	97.5714	89.714	90.831	97.714	96.652
	97.9565	97.1346	91.977	89.714	98.0514	95.429
Mean Value	**97.5257**	**96.353**	**90.8455**	**90.2725**	**97.8827**	**95.9955**
PIMA	81.0156	75.9376	74.479	78.125	80.729	76.823
	79.4583	75.1302	76.823	77.604	74.74	79.167
Mean Value	**80.2349**	**75.5309**	**75.651**	**77.8645**	**77.7345**	**77.995**
BUPA	75.3488	70.1745	70.349	68.208	72.093	74.566
	76.9368	68.1502	71.676	65.698	71.512	73.41
Mean Value	**76.1428**	**69.1476**	**71.0125**	**66.953**	**71.8025**	**73.988**
WINE	96.46065	98.3707	80.899	84.27	98.876	96.629
	93.31455	98.3145	76.404	85.393	100	95.506
Mean Value	**94.8876**	**98.3426**	**78.6515**	**84.8315**	**99.438**	**96.0675**

Figure 4. Performance of proposed method vs. RBF in training set

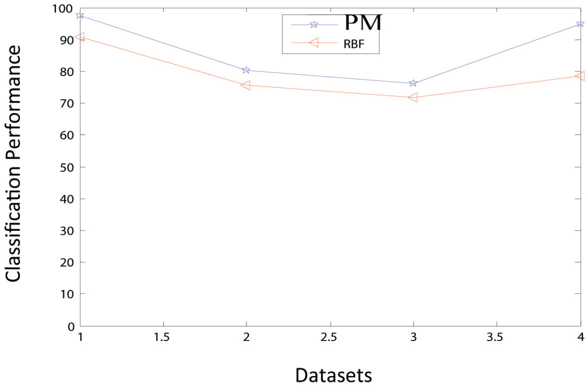


Table 1 presents a summary of the main features of each dataset that has been used in this study.

Methodology and Classification Performance

For the case of the Wisconsin Breast Cancer, PIMA Indians Diabetes and BUPA Liver Disorders datasets, the total set of patterns was randomly divided into two equal parts (database1.dat and database2.dat). Each of these two sets was alternately used either as a training set or as a test set. As we use a stochastic method for data classification, result may vary from simulation to simulation. Each set has been simulated for 50 times. As far as the parameter is concerned we selected the neighborhood size over different iterations based on the following equation.

Figure 5. Performance of proposed method vs. RBF in test set

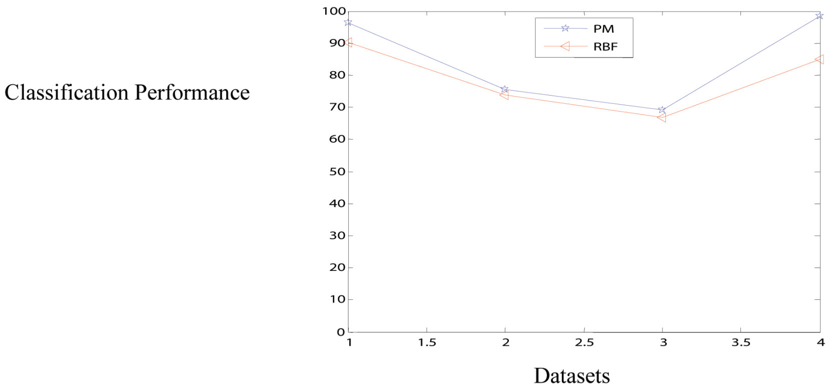
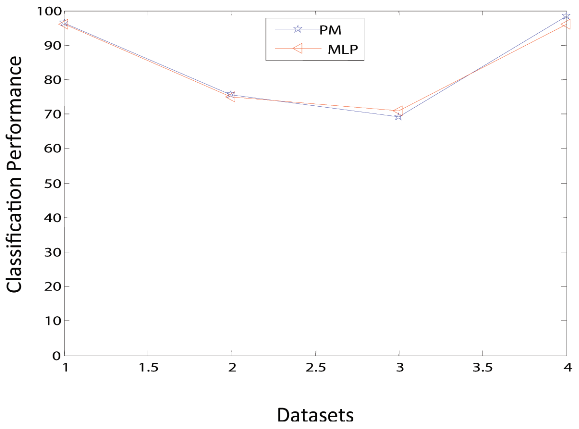


Figure 6. Hit percentage of training set performance comparison of proposed method with MLP



$$nb(t) = \begin{cases} |\, swarm \,| - t & if\; t < |\, swarm \,| - 2 \\ 2 & otherwise \end{cases},$$

where $|swarm|$ is the size of the swarm and t is denoted as iteration.

Similarly the value of the learning probability τ, can be computed by the following equation. This equation is empirically determined.

$$\tau_i = \frac{1}{e^{10} - 1}\left(0.05.e^{10} + 0.45.e^{\frac{10i-1}{|swarm|-1}} - 0.5\right)$$

The results considered here is the average of 50 simulations. Table 2 summarizes the results obtained in the classification of these four datasets using the proposed method.

From Table 2 one can observed that for the case of WINE dataset on an average hit percentage in the training set is not better than the hit

Figure 7. Hit percentage of test set performance comparison of proposed method with MLP

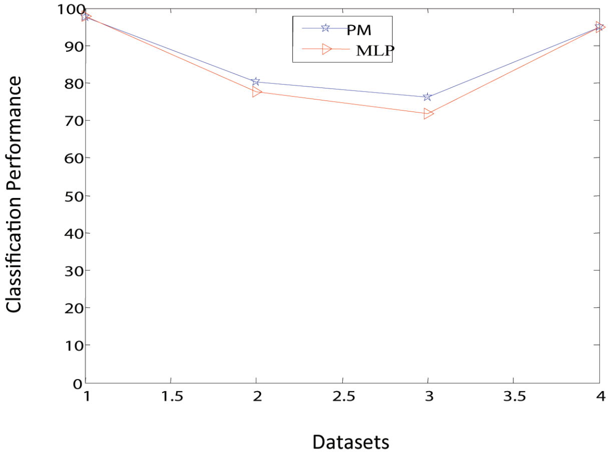


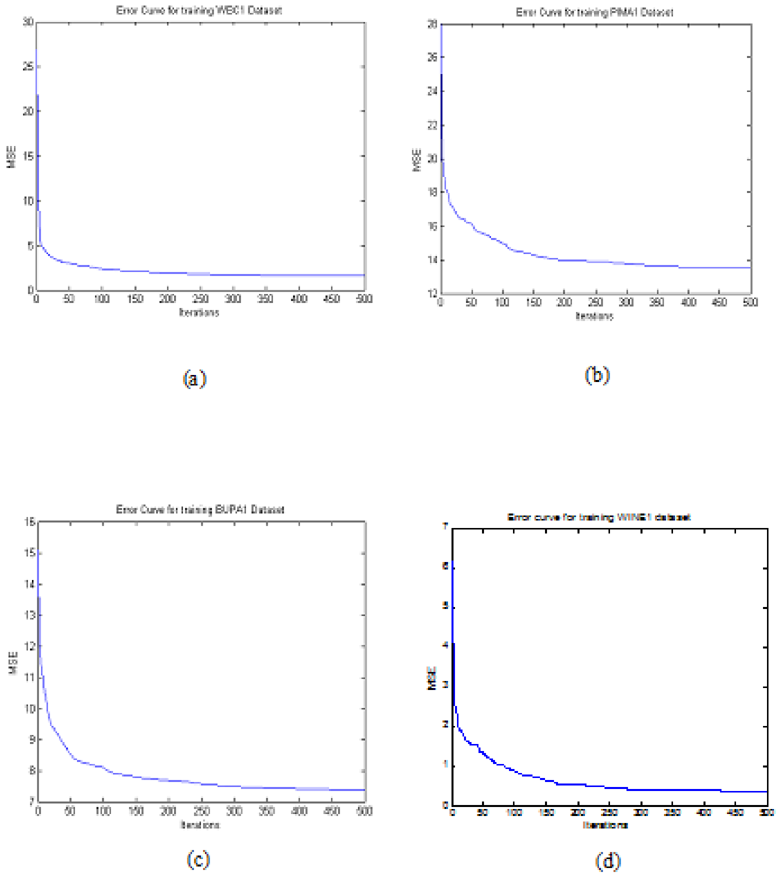
Table 4. Standard deviations obtained from 50 simulations of proposed method

Dataset	Standard Deviation in Training Set	Standard Deviation in Test Set
WBC	0.2256	0.4312
	0.3432	0.3307
Mean Value	0.2844	0.381
PIMA	0.4331	1.2351
	0.7613	0.7287
Mean Value	0.5972	0.9819
BUPA	1.1693	0.9901
	0.4265	1.3746
Mean Value	0.7979	1.1824
WINE	1.424589	0.679323
	1.689402	1.062752
Mean Value	1.5570	0.8710

percentage in test set. This shows that the model required more training even though the classification accuracy of the test cases is better than training cases. In the case of PIMA, WBC and BUPA datasets, on an average, the hit percentage of training set is significantly better than that of test set.

Table 3 shows the comparison of the proposed method with RBF and MLP. The comparative performance of proposed method with RBF is superior, but it is competitive with MLP. Figure 4 shows the performance comparison of proposed method with RBF by considering the hit percentage of training sets. The result obtained from proposed method is outperforming the results obtained from RBF. Similarly, Figure 5 shows the performance comparisons of proposed method with RBF

Figure 8. Error obtained from a) WBC b) PIMA c) BUPA and d) WINE



(a)

(b)

(c)

(d)

by considering the hit percentage of test set. The results are promising, but Figure 6 and 7 shows a very close performance of proposed method and MLP. In other words we can claim the obtained result from proposed method is competitive with MLP in terms of accuracy.

Dependability of the Classifier

In addition to the classification accuracy obtained from a classifier, dependability of a model resides on the consistency of the results obtained. To verify the dependability of the model standard deviation of the results obtained in 50 simulations is taken and presented in Table 4.

Efficiency of the Classifier

Efficiency of the classifier may be evaluated in different ways. Here we present the error curves at Figure 8 for uniform 500 iterations. The rate of convergence of the models clearly depicts the efficiency of the models. The Mean Squared Error (MSE) values have been evaluated from the average of the squared error of 30 particles distributed in the search space.

In this experiment, our objective is to select suitable weights for the network, which can yield better or competitive results by minimizing error. In addition to this we have to ensure that the network should not be over trained. Therefore, the iterations suitable for breast cancer are taken from the range [50-150] and the range of other three dataset is lies between the closed interval [100, 200].

CONCLUSIONS AND FUTURE RESEARCH PATHS

In this paper, we have evaluated the proposed method for the task of classification in data mining. The proposed method expands the given set of inputs into three types of fuzzy sets such as low, medium and high with the Gaussian membership function. These inputs are fed into a simple perceptron. Our novel sequence based global particle swarm optimizer is used for evolving a set of weights. The weight vectors are represented by particles of the swarm and are being spread in a distributed environment. The experimental studies demonstrated that proposed method performs the classification task quite well. The efficiency of the proposed method is competitive in terms of its rate of convergence. The future work includes the study of optimal number of fuzzy sets required to map an input feature into higher dimensions. Tuning of membership functions in this context is leaving a room for further work. The positive results encountered in this paper suggest that additional experiments may provide further insight into the benefits of fuzzy neural networks trained with sequence bound global particle swarm optimizer.

REFERENCES

Abe, S., & Thawonmas, R. (1997). A fuzzy classifier with ellipsoidal regions. *IEEE Transactions on Fuzzy Systems*, *5*(3), 358–368. doi:10.1109/91.618273

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 201-216).

Auckley, J. J., & Hayashi, Y. (1994). Fuzzy neural networks: a survey. *Fuzzy Sets and Systems*, *66*(1), 1–13. doi:10.1016/0165-0114(94)90297-6

Barrera, J., & Coello, C. A. C. (2009). Limiting the velocity in particle swarm optimization using a geometric series. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (pp. 1739-1740).

Bellman, R., Kalaba, K., & Zadeh, L. A. (1996). Abstraction and pattern classification. *Journal of Mathematical Analysis and Applications*, *13*(1), 1–7. doi:10.1016/0022-247X(66)90071-0

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Gloucestershire, UK: Clarendon Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York, NY: Springer.

Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases*. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, CA: Wordsworth.

Chakraborty, D., & Pal, N. R. (2004). A neuro fuzzy scheme for simultaneous feature selection and fuzzy rule based classification. *IEEE Transactions on Neural Networks*, *15*(1), 110–123. doi:10.1109/TNN.2003.820557

Chen, S. J., & Chen, S. M. (2005). Fuzzy information retrieval based on geometric mean averaging operators. *Computers & Mathematics with Applications (Oxford, England)*, *49*(7-8), 1213–1231. doi:10.1016/j.camwa.2004.06.035

Chen, T. (2011). Applying fuzzy and neural approach for forecasting the foreign exchange rate. *International Journal of Fuzzy System Applications*, *1*(1), 36–48. doi:10.4018/ijfsa.2011010103

Chen, T., Wu, H.-C., & Wang, Y.-C. (2009). Fuzzy neural approaches with example post classification for estimating job cycle time in a wafer fab. *Applied Soft Computing*, *9*, 1225–1231. doi:10.1016/j.asoc.2009.03.006

Craven, M. W., & Shavlik, J. W. (1997). Using neural networks for data mining. *Future Generation Computer Systems*, *13*, 211–229. doi:10.1016/S0167-739X(97)00022-8

Czogala, E., & Leski, J. M. (2000). *Fuzzy and neuro-fuzzy intelligent systems*. Heidelberg, Germany: Physica-Verlag.

Dasarathy, B. S. (1990). *Nearest neighbor pattern classification techniques*. Los Alamitos, CA: IEEE Press.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. New York, NY: Wiley.

Edelstein, H. (1996). Mining data warehouses. *InformationWeek, 48*(4), 561.

Gelman, A., Carlin, J., Stern, H., & Rubin, D. (1995). *Bayesian data analysis*. Boca Raton, FL: Chapman and Hall/CRC.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Han, J., & Kamber, M. (2001). *Data Mining: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann.

Haykin, S. (1999). *Neural Networks: A comprehensive foundation*. Upper Saddle River, NJ: Pearson Education.

Herbrich, R. (2001). *Learning Kernel Classifiers: Theory and Algorithms*. Cambridge, MA: MIT Press.

Hong, W.-S., Chen, S.-J., Wang, L.-H., & Chen, S.-M. (2007). A new approach for fuzzy information retrieval based on weighted power mean averaging operators. *Computers & Mathematics with Applications (Oxford, England)*, *53*(1), 1800–1819. doi:10.1016/j.camwa.2006.04.033

Ishibuchi, H., Nakashima, T., & Murata, T. (2001). Three objective genetic based machine learning for linguistic rule extraction. *Information Science*, *136*(1-4), 109–133. doi:10.1016/S0020-0255(01)00144-X

Jains, A., & Dubes, R. (1998). *Algorithms for clustering data*. Upper Saddle River, NJ: Prentice Hall.

Kennedy, J., & Eberhart, R. (2001). *Swarm intelligence*. San Francisco, CA: Morgan Kaufmann.

Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of the Congress on Evolutionary Computation*.

Kuncheva, L. I. (2002). Switching between selection and fusion in combining classifiers: an experiment. *IEEE Transactions on Systems . Man and Cybernetics-Part B*, *32*(2), 146–156. doi:10.1109/3477.990871

Liang, J. J., Qin, A. K., Suganathan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*(3), 281–296. doi:10.1109/TEVC.2005.857610

Luo, X., Hou, W., & Wang, Z. (2007). Fuzzy neural network model for predicting clothing thermal comfort. *Computers & Mathematics with Applications (Oxford, England)*, *53*(1), 1840–1846. doi:10.1016/j.camwa.2006.10.035

McLachlan, G. J. (2004). *Discriminant analysis and statistical pattern recognition*. Chichester, UK: Wiley-Interscience.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*. New York, NY: Springer.

Michie, D., Spiegelhalter, D., & Taylor, C. (1994). *Machine learning, neural and statistical classification*. Chichester, UK: Ellis Horwood.

Nauck, D., & Kruse, R. (1997). A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems*, *89*(3), 277–288. doi:10.1016/S0165-0114(97)00009-2

Ozcan, E., & Mohan, C. (1998). Analysis of a simple particle swarm optimization system. *Intelligent Engineering Systems through Artificial*. *Neural Networks*, 253–258.

Park, B. J. (2002). Fuzzy polynomial neural networks: hybrid architectures of fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, *10*(5), 607–621. doi:10.1109/TFUZZ.2002.803495

Parrot, D., & Li, X. (2006). Location and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, *10*(4), 440–458. doi:10.1109/TEVC.2005.859468

Piatetsky-Shapiro, G., Symith, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Menlo Park, CA: AAAI Press.

Quinlan, J. R. (1992). *C4.5: Programs for machine learning*. San Francisco, CA: Morgan Kaufmann.

Robert, C. P. (2001). *The Bayesian choice*. New York, NY: Springer.

Roubos, J. A., Setnes, M., & Abony, J. (2003). Learning fuzzy classification rules from labeled data. *Information Sciences*, *150*(1-2), 77–93. doi:10.1016/S0020-0255(02)00369-9

Shi, Y., & Eberhart, R. (1998). Parameter selection in particle swarm optimization. In *Proceedings of the 7th Annual Conference on Evolutionary Computation* (pp. 591-601).

Shi, Y., & Eberhart, R. (1999). Empirical study of particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation* (pp. 1945-1950).

Theodoridis, S., & Koutroumbas, K. (2006). *Pattern recognition*. Amsterdam, The Netherlands: Elsevier.

Uebele, V., Abe, S., & Lan, M. S. (1995). A neural network based fuzzy classifier. *IEEE Transactions on Systems, Man, and Cybernetics*, *25*(2), 353–361. doi:10.1109/21.364829

Watanabe, S., Furuhashi, T., Obata, K., & Uchikawa, Y. (1993). A study on feature extraction using a fuzzy net for off-Line signature recognition. In *Proceedings of the Joint International Conference on Neural Networks* (pp. 2857-2860).

Winkler, R. L. (2003). *Introduction to Bayesian inference and decision*. Gainesville, FL: Probabilistic.

Wu, H.-K., Hsieh, J.-G., Lin, Y.-L., & Jeng, J.-H. (2010). On maximum likelihood fuzzy neural networks. *Fuzzy Sets and Systems*, *161*(21), 2795–2807. doi:10.1016/j.fss.2010.06.003

Zhou, E., & Khotanzad, A. (2007). Fuzzy classifier design using genetic algorithms. *Pattern Recognition*, *40*, 3401–3411. doi:10.1016/j.patcog.2007.03.028