Pattern Recognition using Soft Computing Approaches

Tirtharaj Dash

May, 2014

Department of Comp. Sc. & Engg. and Information Technology

Veer Surendra Sai University of Technology (VSSUT)

Burla, Sambalpur, Odisha–768018, India

Pattern Recognition using Soft Computing Approaches

Thesis submitted in partial fulfillment of the requirement of the degree of

Master of Technology

in

Computer Science and Engineering
(Specialization: Computer Science and Engineering)

by,

Tirtharaj Dash
(Regd. No. 12040085)

Under the supervision of

Dr. H.S. Behera
Reader

May, 2014

Department of Comp. Sc. & Engg. and Information Technology

Veer Surendra Sai University of Technology (VSSUT)

Burla, Sambalpur, Odisha–768018, India

ACKNOWLEDGEMENTS

I take this immense opportunity to express my sincere gratitude towards my supervisor and mentor ***Dr. H.S Behera****, Reader, VSSUT Burla for his invaluable guidance. His profound knowledge, inventive ideas and timely suggestions has nourished this thesis work at every stage. 'Learning how to do research' from him is being a life time experience in itself. It has been a very enlightening and enjoyable experience to work with him. It would have never been possible for me to complete this thesis work without his constant support and encouragement. His support and determination has truly inspired me to aim higher and put my best efforts.*

I extend my heartfelt gratitude to ***Dr. M.R. Kabat****, Head of the Department, Dept. of Comp. Sc. & Engg. and Information Technology, VSSUT Burla, who has left no stone upturned in rendering all the available facilities and guidance in the time of my need. His encouragement and support has proved indispensable during the tenure of my study. I would also like to thank* ***Prof. Rakesh Mohanty****,* ***Prof. (Dr.) C.R. Tripathy*** *for constantly encouraging me to aim high in life.*

I especially indebted to my parents for their love sacrifice and support. They are my first teacher after I came to this world and have set great examples for me about how to live, study and work. I also want to express my love to my sister, ***Juli****, who is the best friend in my life. I would like to thank* ***Soumya Ranjan Sahu****,* ***Nayan K.S. Behera****,* ***Rakesh Ranjan Swain****,* ***Priyabrata Dash****,* ***Nikunja Bihari Kar****, my roommates* ***Pritish Kumar Behera*** *and* ***Gokula Chandra Sethi*** *for being constant supporters for me during these two years of M.Tech study.*

At last but not least I would like to express my gratefulness to all those who either directly or indirectly helped me in successful completion of my thesis work.

Tirtharaj Dash

Regd. No. 12040085

M.Tech. (Comp. Sc. & Engg.)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In case of decision making problems, classification of pattern is a complex and crucial task. Pattern classification using multilayer perceptron (MLP) trained with back propagation learning becomes much complex with increase in number of layers, number of nodes and number of epochs and ultimately increases computational time. As a part of contribution, an attempt has been made to use Fuzzy MLP and its learning algorithm for pattern classification. The time complexities of the algorithm have been analyzed. A comparison of training performance has been done between MLP and the Fuzzy MLP by considering six cases. A new performance evaluation factor 'convergence gain' has been introduced for performance evaluation. Furthermore, Genetic algorithm (GA) and two swarm-based heuristic algorithms such as Particle swarm optimization (PSO) and Gravitational search algorithm (GS). A comparative study on classification has been conducted using seven different datasets. A new hybrid training model, called GSPSO, has been developed by combining two metaheuristics, gravitational search (GS) and particle swarm optimization (PSO). The proposed model combines local searching technique of GS with social movement of PSO. The GSPSO model has been used with Fuzzy Multilayer Perceptron for medical data classification. Five medical datasets from UCI machine learning repository are employed for evaluating the performance of the proposed GSPSO based Fuzzy MLP (Fuzzy MLP-GSPSO) model. The experimental results show that Fuzzy MLP-GSPSO model outperforms Fuzzy MLP-GS and Fuzzy MLP-PSO for all tested cases.

Key terms: pattern, classification, multilayer perceptron, Fuzzy multilayer perceptron, UCI dataset, Genetic algorithm, Particle swarm optimization, Gravitational search, classification accuracy

Chapter-1

1

Introduction

Preview:

This chapter introduces the reader to basic concepts of data classification, pattern recognition, soft computing models, Artificial Neural Network (ANN) which will help in understanding rest chapters.

1.1. Pattern Recognition

Pattern recognition is the scientific discipline whose goal is to classify objects into a number of categories or classes. Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified. We will refer to these objects using the generic term patterns [1,2]. Pattern recognition has a long history, but before the 1960s it was mostly the output of theoretical research in the area of statistics. As with everything else, the advent of computers increased the demand for practical applications of pattern recognition, which in turn set new demands for further theoretical developments. As our society evolves from the industrial to its postindustrial phase, automation in industrial production and the need for information handling and retrieval are becoming increasingly important. This trend has pushed pattern recognition to the high edge of today's engineering applications and research. Pattern recognition is an integral part of most machine intelligence systems built for decision making. Pattern recognition is studied in many fields, including psychology, psychiatry, ethology, cognitive science, traffic flow and computer science.

1.2. Data classification

Data Classification and **Pattern recognition** are nearly synonymous with machine learning and usually used interchangeably. This branch of artificial intelligence focuses on the recognition of patterns and regularities in data. In many cases, these patterns are learned from labeled 'training' data (supervised learning), but when no labeled data is available other algorithms can be used to discover previously unknown patterns (unsupervised learning).

The terms pattern recognition, machine learning, data mining and knowledge discovery in databases (KDD) are hard to separate, as they largely overlap in their scope. Machine learning is the common term for supervised learning methods and originates from artificial intelligence, whereas KDD and data mining have a larger focus on unsupervised methods and stronger connection to business use. Pattern recognition has its origins in engineering, and the term is popular in the context of computer vision: a leading computer vision conference is named Conference on Computer Vision and Pattern Recognition. In pattern recognition, there may be a higher interest to formalize, explain and visualize the pattern; whereas machine learning traditionally focuses on maximizing the recognition rates. Yet, all of these domains have evolved substantially from their roots in artificial intelligence, engineering and statistics; and have become increasingly similar by integrating developments and ideas from each other [3,4].

In machine learning, pattern recognition is the assignment of a label to a given input value. In statistics, discriminant analysis was introduced for this same purpose in 1936. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of classes (for example, determine whether a given email is "spam" or "non-spam"). However, pattern recognition is a more general problem that encompasses other types of output as well. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); and parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence.

Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to perform *most likely* matching of the inputs, taking into account their statistical variation. This is opposed to pattern matching algorithms, which look for exact matches in the input with pre-existing patterns. A common example of a pattern-matching algorithm is regular expression matching, which looks for patterns of a given sort in textual data and is included in the search capabilities of many text editors and word processors. In contrast to pattern recognition, pattern matching is generally not considered a type of machine learning, although pattern-matching algorithms (especially with fairly general, carefully tailored patterns) can sometimes succeed in providing similar-quality output to the sort provided by pattern-recognition algorithms. Pattern recognition problems are more accurately solved by employing soft computing techniques such as neural networks, fuzzy logic, evolutionary computing etc.

1.3. Soft Computing

Soft Computing is a term used in computer science to refer to problems in computer science whose solutions are unpredictable, uncertain and between 0 and 1. Soft Computing became a formal area of study in Computer Science in the early 1990s [5]. Earlier computational approaches could model and precisely analyze only relatively simple systems. More complex systems arising in biology, medicine, the humanities, management sciences, and similar fields often remained intractable to conventional mathematical and analytical methods. That said, it should be pointed out that simplicity and complexity of systems are relative, and many conventional mathematical models have been both challenging and very productive. Soft computing deals with imprecision, uncertainty, partial truth, and approximation to achieve practicability, robustness and low solution cost. As such it forms the basis of a considerable amount of machine learning techniques [6]. Recent trends tend to involve evolutionary and swarm intelligence based algorithms and bio-inspired computation.

There is a main difference between soft computing and possibility. Possibility is used when we don't have enough information to solve a problem but soft computing is used when we don't have enough information about the problem itself. These kinds of problems originate in the human mind with all its doubts, subjectivity and emotions; an example can be determining a suitable temperature for a room to make people feel comfortable.

Components of soft computing include Neural networks (NN), Perceptron, Support Vector Machines (SVM), Fuzzy logic (FL), Evolutionary computation, probability theory, chaos theory etc. Evolutionary computing includes Genetic algorithms, Differential evolution, Metaheuristic and Swarm Intelligence, Ant colony optimization, Particle swarm optimization, Firefly algorithm, Cuckoo search, Gravitational search, Flower pollination etc.

Generally speaking, soft computing techniques resemble biological processes more closely than traditional techniques, which are largely based on formal logical systems, such as sentential logic and predicate logic, or rely heavily on computer-aided numerical analysis (as in finite element analysis). Soft computing techniques are intended to complement each other [6]. Unlike hard computing schemes, which strive for exactness and full truth, soft computing techniques exploit the given tolerance of imprecision, partial truth, and uncertainty for a particular problem. Another common contrast comes from the observation that inductive reasoning plays a larger role in soft computing than in hard computing.

1.4. Artificial Neural Networks

In computer science and related fields, artificial neural networks (ANNs) are computational models inspired by an animal's central nervous systems (in particular the brain) which are capable of machine learning as well as pattern recognition. Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs. For example, a neural network for handwriting recognition is defined by a set of input neurons which may be activated by the pixels of an input image. The activations of these neurons are then passed on, weighted and transformed by a function determined by the network's designer, to other neurons. This process is repeated until finally, an output neuron is activated. This determines which character was read. Like other machine learning methods, systems that learn from data, neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition.

1.4.1. Evolution of Neural Network

The examination of the central nervous system was the inspiration of neural networks. In an Artificial Neural Network, simple artificial nodes, known as 'neurons', 'neurodes', 'processing elements' or 'units', are connected together to form a network which mimics a biological neural network.

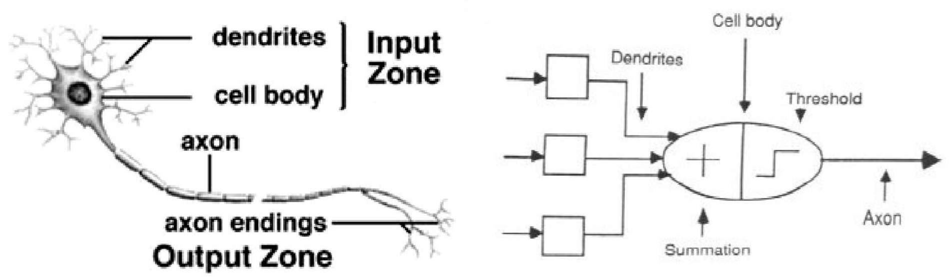


Figure 1.1. Similarities between biological neuron and artificial neural network

Neural networks are similar to biological neural networks in performing functions collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned. The term 'neural network' usually refers to models employed in statistics, cognitive psychology and artificial intelligence. Neural network models which emulate the central nervous system are part of theoretical neuroscience and computational neuroscience. Figure 1.1 given above shows the similarities between biological neuron and artificial neural net.

In modern software implementations of artificial neural networks, the approach inspired by biology has been largely abandoned for a more practical approach based on statistics and signal processing. In some of these systems, neural networks or parts of neural networks (like artificial neurons) form components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such systems is more suitable for real-world problem solving, it has little to do with the traditional artificial intelligence connectionist models. What they do have in common, however, is the principle of non-linear, distributed, parallel and local processing and adaptation. Historically, the use of neural networks models marked a paradigm shift in the late eighties from high-level (symbolic) artificial intelligence, characterized by expert systems with knowledge embodied in if-then rules, to low-level (sub-symbolic) machine learning, characterized by knowledge embodied in the parameters of a dynamical system.

1.4.2. Learning of ANN

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning. It will be better to understand each learning paradigm with suitable examples. Examples of tasks falling in each category of learning are given below.

(a) Supervised learning

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a ‘teacher’, in the form of a function that provides continuous feedback on the quality of solutions obtained so far.

(b) Unsupervised learning

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

(c) Reinforcement learning

ANNs are frequently used in reinforcement learning as part of the overall algorithm. Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

1.5. Objective of the Thesis

The objective of the thesis would be to design and develop pattern recognition and data classification algorithms that can be effectively and efficiently employed for various pattern recognition problems in the real world. Major objective is to test the developed and newly proposed algorithms with a number of datasets so that the proposed models will comply with the real world scenario. The advantages of the implementations in the thesis could be as follows.

- The works are interdisciplinary i.e. proposed models can be easily employed in interdisciplinary research areas
- Least manual overheads i.e. everything will be automated
- Black box in nature i.e. user does not have to worry about the back ground details of the models
- Classification and pattern recognition accuracy along with decision speed are of major performance evaluation parameters

1.6. Thesis organization

This thesis is composed of five chapters. The first chapter introduces the reader to basic concepts of data classification, pattern recognition, soft computing models, Artificial Neural Network (ANN) without going much into the depth. Second chapter presented a proposed pattern recognition model combining fuzzy logic with ANN, called Fuzzy Multilayer Perceptron (Fuzzy MLP) along with its performance evaluation. Third chapter presents three training algorithms for Fuzzy MLP based on three metaheuristics, Genetic algorithm (GA), Particle Swarm Optimization (PSO) and Gravitational search technique (GS). In chapter 4, a hybrid of GS and PSO, called GSPSO, has been proposed which improves the performance of all the algorithms given in chapter 3. Proposed GSPSO algorithm has been tested for medical datasets and corresponding performances of the algorithms are compared for each datasets. Chapter-5 concludes the thesis followed by set of publications which are output of this thesis work.

References

[1] C.M. Bishop, “Neural Networks for Pattern Recognition”, Oxford University Press, Oxford, UK, 1995.

[2] A.Abraham, “Meta-Learning Evolutionary Artificial Neural Networks,” Neurocomputing Journal, Vol. 56c, Elsevier Science, Netherlands, (1–38), 2004.

[3] T.J. Ross, “Fuzzy logic with engineering applications”, 2nd edn., John Wiley & Sons, ltd, ISBN: 0-470-86075-8.

[4] J.H. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, Ann Arbor, Michigan; re-issued by MIT Press (1992).

[5] L.A. Zadeh, “Fuzzy Logic, Neural Networks, and Soft Computing,” Communication of the ACM, March 1994, Vol. 37 No. 3, pages 77-84.

[6] D. K. Chaturvedi, “Soft Computing: Techniques and Its Applications in Electrical Engineering,” Springer, (2008).

Chapter-2

2

A Fuzzy MLP Approach for Non-linear Pattern Classification

Preview:

This chapter presents a detailed description of development of a Fuzzy Multilayer Perceptron for non-linear pattern classification problems. Subsections in the chapter elaborate related works, methodology, and performance analysis of the said approach along with a conclusion section.

2.1. Motivation

Pattern Classification has been one of the most frequently needed tasks in different fields of science and engineering studies [1]. In real world, every object has certain set of attributes. These attributes help in keeping that object belong to a certain group, called **class** and an object in a particular group follows a certain pattern. A pattern classification problem can be defined as a problem where an object is assigned into a predefined group or class based on a set of observed attributes related to that object [1]. For example, we classify the programming languages as high-level or low-level, object-oriented or object-based depending on a set of properties possessed by these languages such as syntax, semantics, compilation procedure etc. Similarly, several other problems in business, science, industry, and medicine can be treated as classification problems. Examples include bankruptcy prediction [2], stock market forecasting [2], credit scoring, medical diagnosis [3][10], medical data classification [3], power quality control and classification [4][5], handwritten character recognition [6], signature verification [7], fingerprint recognition [8], speech recognition [9] etc.

Finding solutions to such classification problems has been a crucial research area in the field of technology. Recent researches have used either statistical techniques or data mining techniques, such as neural computing. Statistical methods use probability theory, decision theory to classify a set of data; whereas, neural computing is a technique which uses a neural network (NN) model for classifying the input data. With increasing demands of the problems, the NN approach for classifying pattern is becoming popular and new NN models have been developed in the process. The reason behind this fact is that, the statistical methods use certain set of assumptions and conditions to satisfy the solution. And these models find their application in a few real world problems [1]. This limitation of statistical methods has been reduced by the use of NN techniques. The first and foremost advantage of using NN techniques is that, NNs are nonlinear which let them adjust to the real world input data without any explicit specification or external conditions. Secondly, they are universal functional approximation models which can approximate any function with arbitrary accuracy. However, NN techniques employes statistical analysis of the processed data to approximate the results.

Significant progress has been made in recent years in the area of pattern recognition and neuro-computing and Feed-forward Multilayer Perceptron (MLP, which is formed by arranging multiple layers of single Perceptrons) is the most widely used NN classifiers for the works. A technique hybrid of fuzzy rules and NN, called neurofuzzy has been applied to model some real world problems such as medical diagnosis [10]. A novel model was developed by Kang and Brown [11] for classifying data. This is an unsupervised NN, called as adaptive function neural network and has no hidden layer in its architecture.

The concept of Fuzzy MLP has been proposed by Mitra et al. [12] for classification and rule generation purposes where the authors applied Fuzzy MLP to extract a set of rules from input dataset and these rules were used to find classification results. This paper focuses on development of a Fuzzy MLP model for nonlinear pattern classification where target class is not a linear function of input attributes. Performance of this model has been compared with that of MLP developed for the same problems. It should be noted that both Fuzzy MLP and MLP nets uses gradient descent learning for knowledge updates during training.

2.2. Proposed Methodology

The proposed Fuzzy Multi-layered Perceptron (Fuzzy MLP) is n-layer architecture, where *$n>3$*. The layers are (i) input layer (IL), (ii) output layer (OL) and (iii) hidden layer (HL). The number of hidden layers may be increased or decreased based on the problem for which the model is developed. It is a feed-forward neural network, where non-linear elements, called neurons are arranged in layers. The data flow from the input layer to the output layer via hidden layer(s) and intermediate processing. Learning is supervised for Fuzzy MLP where the computed error at the output layer acts as the supervisor for different updates during training. A generalized delta rule (Δ-rule) is followed for this process and has been given in following pseudocode. The Δ-rule uses a set of equations as given in Equations 1-6.

The first phase of the learning is the fuzzification of the input data. In this work, Spline based or S-shaped fuzzy membership function (MF) has been used for this phase. This MF puts the input dataset in a range *$[a,b]$*.The equation for S-shaped MF has been given in Equation-7. The architecture of Fuzzy MLP (see Figure 2.1) along with its learning algorithm has been given as follows.

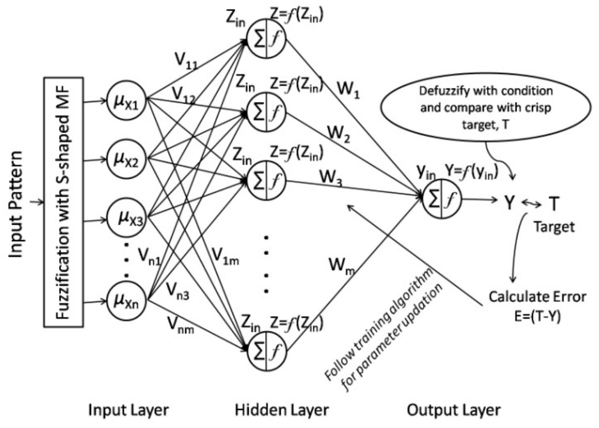


Figure 2.1. Architecture of Fuzzy MLP

In this work binary sigmoid function has been used as the activation function in HL and OL and is defined as $f(x) = 1/(1+e^{-x})$. The weights of the neural network are denoted as follows. Weight between input and hidden layer is denoted as V, weight between hidden and output layer is denoted as W, b_1 is the array of biases in hidden layer, b_2 is bias used in output layer and Δ is used to denote the change in corresponding parameter. In Equations 1-7, Vin_i represents the incoming input-hidden layer weight to the considered node *i* for which the net input Z_{in} is being calculated. It should be noted that all the input, processing and output are carried out with decimal values only.

$$Z_{in} = b_1 + \sum_{i=1}^{n} X_i Vin_i \tag{1}$$

$$y_{in} = b_2 + \sum_{i=1}^{m} Z_i W_i \tag{2}$$

$$\Delta W = \{\alpha \times E \times f'(y_{in}) \times Z_i\} + (\mu \times \Delta W_{old}) \tag{3}$$

$$\Delta b_2 = \alpha \times E \times f'(y_{in}) \tag{4}$$

$$\Delta V = \{\alpha \times E \times f'(y_{in}) \times W \times f'(Z_{in}) \times X\} + (\mu \times \Delta V_{old}) \tag{5}$$

$$\Delta b_1 = \alpha \times E \times f'(y_{in}) \times W \times f'(Z_{in}) \tag{6}$$

$$f(x,a,b) = \left\{ \begin{array}{l} 0, x < a \\ 2\left(\frac{x-a}{b-a}\right)^2, a \le x \le \frac{a+b}{2} \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2, \frac{a+b}{2} \le x \le b \\ 1, x > b \end{array} \right\} \tag{7}$$

***Complexity analysis**:* Due to three layered architecture of Fuzzy MLP, 4 major phases (two forward and two back-propagations) of processing is done. The first forward computation is from IL to HL and it occurs in *O(n×m)* times. The next forward computation is from HL to OL which takes *O(m)* times. The same computation is repeated for back propagation phase also. So, the total time complexity of the proposed algorithm is T(m,n)=2×[O(n×m)+O(m)]. As n×m >> m, therefore T(n,m)=O(n×m).

2.2.1. Pseudo-code for Fuzzy MLP training

```
initialize W, V, $b_1$, $b_2$, $\alpha$, $\mu$, MSE=0
fuzzify the training input pattern using S-shaped MF
input the fuzzified values to the input layer neurons
while (termination criteria is not satisfied)
do
    for each training pattern
        for each hidden neuron
            Calculate input to the hidden neuron ($Z_{in}$)
            Calculate activation of hidden neuron, $Z = f(Z_{in})$
        end for
        Calculate net input to the output neuron ($y_{in}$)
        Calculate activation of output neuron, $Y = f(y_{in})$
        Y=Defuzzify the fuzzy output Y ($\mu_Y$)
        Compute the error, E = T–Y; where T is the corresponding target
        /*Back propagate the error to update weights
        Calculate $\Delta W$
        Update $b_2$
        for each hidden neuron
            Calculate $\Delta V$
            Update $b_1$
            end for
        end for
        update $MSE = (MSE + E^2)/n$
end while
```

2.3. Experimental Details

The proposed algorithm has been evaluated with six public domain datasets from the popular UCI machine learning repository [14]. The following subsection summarizes the datasets.

2.3.1. Dataset Description

The following table shows a summary of datasets showing total number of patterns, total number of attributes and number of distinct target classes. Further details and detailed properties of each dataset can be obtained from [14].

Table 2.1. Dataset properties

Dataset	Number of patterns	Number of attributes (including target attribute)	Number of target classes
Iris	150	5	3
Abalone	4177	9	29
Breast-cancer-Wisconsin (BCW)	699	10	2
Glass	214	10	7
Soybean	47	36	4
Wine	178	14	3

2.3.2. Experimental Setup and Fuzzy MLP parameters

Both MLP and fuzzy MLP are implemented in MATLAB R2010a which is installed in a PC having Windows 32-bit XP professional OS and 2GB of main memory. The processor is Intel dual core and each processor has an equal computation speed of 2 GHz (approx.).

Here Convergence of MSE is considered for performance evaluation of pattern classification using both the NNs (MLP and Fuzzy MLP). A few neural network parameters are extensively considered for this work. Those are (i) number of nodes in hidden layer, (ii) learning rate (α), (iii) momentum factor (μ), (iv) type of learning. These parameters were varied from their minimum to maximum value depending on their priority based on their effect on NN training. The following subsections discuss a few issues related to these parameter settings.

Number of hidden units:

The number of learning steps in this work is high and therefore the training phase has extensive calculation. Therefore, selection of the number of hidden nodes in the network is a problem. If the number of hidden nodes is small, then the patterns to be learnt may not possibly be represented as the network capacity is small. If the number is higher than the number of independent variable of the error function also increases and training time becomes high. Therefore, in this work, the number of hidden units is set relative to the number of input units. Number of nodes in the hidden layer (*m*) is fixed at 3/2 of the number of nodes in input layer (*n*); i.e. *m=3n/2* [13].

Learning rate (α):

A high learning rate leads to rapid learning but the weights of NN may oscillate, whereas a smaller learning rate leads to slower learning of the net. In this work, the learning rate (α) is tuned from its minimum, 0 to its maximum possible value, 1 during the network training. The result hence obtained for each α is noted for each dataset. The values of α are set to 0.05, 0.10, 0.25, 0.40, 0.55, 0.70, 0.85 and 0.99 simultaneously.

Momentum factor (μ):

In the developed Fuzzy MLP, knowledge update (weight change) is a combination of current weight gradient and the previous gradient. This method is suitable when some training data are very different from a majority of the data. Sometimes, a small learning rate is used to avoid any disruption of the direction of learning when very unusual pair of training pattern is presented. And in this work, one cannot make certain about the same as the cases are real world cases. Therefore, a momentum factor (μ) is added to the weight update formula to avoid any kind of disruption and make the error to converge faster. In past researches, μ has been set to 0.50 as the optimal value [13]. Therefore, in this work, μ is fixed to 0.50 for all the experiments.

Type of learning:

In this case, batch learning is preferred because the weight update is a contribution of whole set of patterns. The weight change for each epoch may be computed as given in Equation-8. In this work, number of epoch is fixed to 100.

$$\Delta W = \frac{1}{P}\sum_{i=1}^{P} \Delta W_p \tag{8}$$

2.3.3. Experimental Results

As mentioned earlier, the results obtained by employing both fuzzy-MLP and MLP algorithms are compared for each datasets. In this paper, convergence of MSE is considered for performance evaluation of pattern classification using both the algorithms. However, the learning parameters are varied from their minimum to maximum value and the results are noted. Number of nodes in the hidden node has set to 3/2 of the number of nodes in input layer; i.e. $3 \times n/2$. In this work, the learning rate (α) is the most crucial parameter which affects the learning process of the NN. So, it is varied from 0 to 1 in the learning process and a result for each α is noted for each dataset as shown in next subsections. The typical values of α those were chosen are 0.05, 0.10, 0.25, 0.40, 0.55, 0.70, 0.85 and 0.99. The momentum factor (μ) is set to 0.50 for all the experiments. As the fuzzy-MLP is converging very much faster than the MLP net, the number of epoch is set to 100. A table has been maintained for each datasets to show the minimum MSE obtained by MLP and fuzzy-MLP. The table also shows the time consumed by both the algorithms for the process of pattern classification. To check the efficiency of fuzzy-MLP algorithm over classical MLP model, a factor called convergence gain (C_g) has been introduced and is defined in equation-9.

$$C_g = \frac{Min\_MSE_{MLP} - Min\_MSE_{Fuzzy-MLP}}{Min\_MSE_{MLP}} \tag{9}$$

2.3.3.1.Case-1 (IRIS dataset)

The training performance of the fuzzy-MLP and MLP algorithms are evaluated with Iris dataset. The result summary has been given in Table 2.2. It should be noted that the simulation time presented in the table is an average of 5 executions. As number of input attributes for Iris dataset is 4, the number of neurons in hidden layer is set to 6. Results are shown for 100 epochs and μ is set to 0.50. It is clear from the above table that when α is set to 0.99, the propose fuzzy-MLP algorithm is converging to a minimum error of 0.016 (approx.) within 100 epochs only where as the MLP algorithm is converging only to 1.67 (approx.) with this number of epochs. The convergence gain is approximately 99% when α is set to 0.99. However, the overall performance of the proposed algorithm is better than the classical MLP algorithm when the convergence gain is considered. The simulation time of the proposed algorithm is also lesser than that of MLP algorithm in almost all α settings. A plot has been given in Figure 2.2 by taking number of epoch in X-axis and MSE in Y-axis to illustrate the convergence of error in fuzzy-MLP algorithm for each α. The Figure also shows that α = 0.99 is the best value for pattern classification in Iris dataset. However, α can also be set to 0.85 for which the result of convergence is approximately equal to that in α = 0.99.

2.3.3.2. Case-2 (ABALONE dataset)

The result summary for Abalone dataset has been given in Table 2.3. Here number of hidden units is equal to 12.Unlike results in Iris dataset, the minimum MSE obtained by MLP and fuzzy-MLP algorithms are having small difference of 0.39. The gain also shows that the later one is not showing better efficiency as compared to that in the Iris dataset. The plot given in Figure 2.3 shows the convergence of error when α is set to the different values. It can be seen from Table-2.3 and the above given plot that the convergence of error is hardly depending on the learning rate for the Abalone dataset. However, with α=0.05 and 0.10, the error is converging to its minimum after 15 epochs, where as for other α values it's getting to its minimum within 10 epochs.

Table 2.2 Performance of fuzzy-MLP for Iris dataset

α	Minimum MSE		Convergence	Simulation Time (sec)	
	MLP	Fuzzy-MLP	Gain (C_g)	MLP	Fuzzy-MLP
0.05	1.66686	0.07183	0.9569	62.82	55.82
0.10	1.66676	0.04323	0.9741	62.77	55.79
0.25	1.66670	0.03665	0.9780	56.06	50.55
0.40	1.66669	0.02718	0.9837	57.05	56.70
0.55	1.66668	0.02175	0.9870	58.96	47.94
0.70	1.66668	0.01873	0.9888	55.18	56.11
0.85	1.66667	0.01687	0.9899	60.62	49.46
0.99	1.66667	0.01566	0.9906	60.81	46.78

Table 2.3 Performance for for Abalone dataset

α	Minimum MSE		Convergence Gain (C_g)	Simulation Time (sec)	
	MLP	Fuzzy-MLP		MLP	Fuzzy-MLP
0.05	0.5105	0.1045	0.7952	1832.82	1844.7
0.10	0.5024	0.1034	0.7942	838.68	885.39
0.25	0.4952	0.1035	0.7909	866.88	863.60
0.40	0.4894	0.1035	0.7885	846.67	841.34
0.55	0.4843	0.1035	0.7862	842.05	838.90
0.70	0.4815	0.1036	0.7848	841.62	840.23
0.85	0.4794	0.1037	0.7836	834.07	847.16
0.99	0.4780	0.10394	0.7826	824.41	843.72

Table 2.4 Performance for BCW dataset

α	Minimum MSE		Convergence Gain (C_g)	Simulation Time (sec)	
	MLP	Fuzzy-MLP		MLP	Fuzzy-MLP
0.05	3.75824	0.04232	0.9887	152.17	156.42
0.10	3.75823	0.04122	0.9890	154.13	158.57
0.25	3.75823	0.04128	0.9890	156.24	149.07
0.40	3.75822	0.04150	0.9890	148.39	151.29
0.55	3.75822	0.04169	0.9889	156.81	150.67
0.70	3.75822	0.04183	0.9889	147.51	147.36
0.85	3.75822	0.04195	0.9888	149.75	148.83
0.99	3.75822	0.04207	0.9888	153.26	153.62

2.3.3.3.Case-3 (BCW dataset)

The result summary for BCW dataset has been given in Table 2.4. Here number of hidden units is equal to 14. The fuzzy-MLP algorithm outperforms the classical MLP algorithm with a convergence gain factor of approximately 98%. As the BCW dataset is large, it can be proposed that the proposed algorithm can be applied to larger datasets in real life scenarios. A plot has been given in Figure 2.4 shows the convergence of error when α is set to the different values.

2.3.3.4. Case-4 (GLASS dataset)

The result summary for Glass dataset has been given in Table 2.5. Here number of hidden units is set to 15, as the number of input attributes is 10. It can be seen from the table that the maximum gain obtained is 99% when α is set within a range of 0.25-0.99. The plot given in Figure 2.5 shows the convergence of error when α is set to the different values between 0 and 1.

2.3.3.5. Case-5 (SOYBEAN dataset)

The result summary for Soybean dataset has been given in Table 2.6. Here number of hidden units is equal to 51. It can be noted that the Soybean dataset is the dataset is having maximum input characteristic, i.e. 34. And the average gain obtained is approximately 95%. Therefore, the proposed algorithm can be applied to problems with larger predicting attributes. The plot given in Figure 2.6 shows the convergence of error when α is set to the different values. It can be seen from Figure 2.5 that, the MSE obtained by the employing proposed fuzzy-MLP algorithm is converging to their corresponding minimum with different α after 25 epochs. The plots with *α=0.05 and α=0.25* are not converging within 100 epochs. However, observing the plots, the converging is inversely proportional to α. That means the result with maximum α is converging with minimum epoch and result with minimum α is converging with more number of epochs.

2.3.3.6. Case-6 (WINE dataset)

The result summary for Wine dataset has been given in Table 2.7. Here number of hidden units is equal to 18. Observing the MSEs obtained by fuzzy-MLP algorithm with α set to 0.05 and 0.10 there is a difference of 0.27. The plot given in Figure 2.7 shows the convergence of error when α is set to the different values. The convergence obtained with the Wine dataset is similar to that of Glass dataset where the MSE plot is gradually increasing after 40 epochs.

Table 2.5 Performance for Glass dataset

α	Minimum MSE		Convergence	Simulation Time (sec)	
	MLP	Fuzzy-MLP	Gain (C_g)	MLP	Fuzzy-MLP
0.05	7.57481	0.12105	0.9840	44.92	53.37
0.10	7.57479	0.09537	0.9874	44.53	48.71
0.25	7.57478	0.07441	0.9902	43.56	41.70
0.40	7.57476	0.06987	0.9908	43.81	41.64
0.55	7.57474	0.06626	0.9913	42.39	42.81
0.70	7.57472	0.06396	0.9916	42.30	45.66
0.85	7.57471	0.06217	0.9918	44.64	43.78
0.99	7.57471	0.06067	0.9920	42.69	41.59

Table 2.6 Performance for Soybean dataset

α	Minimum MSE		Convergence	Simulation Time (sec)	
	MLP	Fuzzy-MLP	Gain (C_g)	MLP	Fuzzy-MLP
0.05	4.31924	0.35186	0.9185	10.68	11.65
0.10	4.31922	0.35183	0.9185	12.06	12.08
0.25	4.31902	0.35131	0.9187	11.09	11.55
0.40	4.31918	0.10119	0.9766	10.99	9.43
0.55	4.31918	0.03035	0.9930	9.61	9.20
0.70	4.31917	0.20958	0.9515	9.38	9.83
0.85	4.31917	0.02967	0.9931	10.17	10.19
0.99	4.31916	0.02975	0.9931	10.38	10.16

Table 2.7 Performance for Wine dataset

α	Minimum MSE		Convergence	Simulation Time (sec)	
	MLP	Fuzzy-MLP	Gain (C_g)	MLP	Fuzzy-MLP
0.05	1.47754	0.36559	0.7526	37.71	36.65
0.10	1.47754	0.08701	0.9411	37.73	42.44
0.25	1.47753	0.06913	0.9532	37.40	35.42
0.40	1.47753	0.06440	0.9564	35.91	40.93
0.55	1.47753	0.06271	0.9576	35.34	36.01
0.70	1.47753	0.06352	0.9570	36.99	36.90
0.85	1.47753	0.06679	0.9548	39.09	39.30
0.99	1.47753	0.06226	0.9579	39.81	40.36

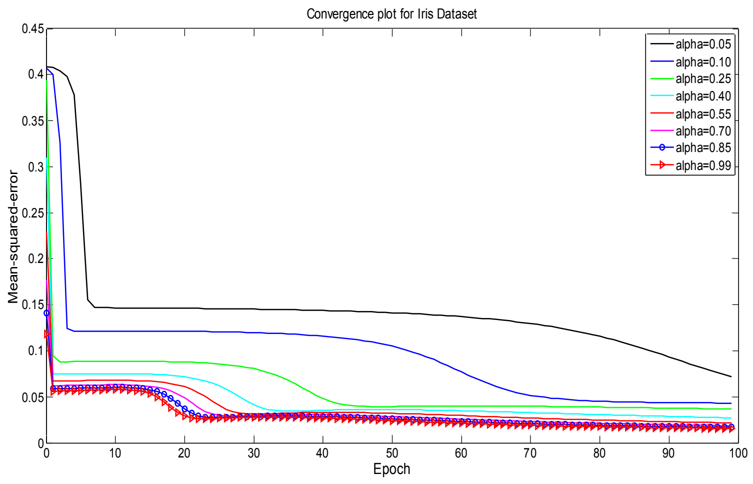


Figure. 2.2 Convergence plot obtained by employing Fuzzy MLP algorithm for Iris dataset

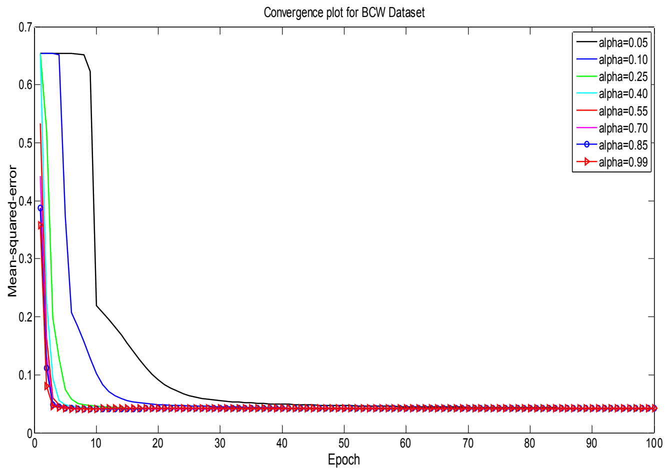


Figure. 2.3. Convergence plot obtained by employing fuzzy-MLP algorithm for BCW dataset

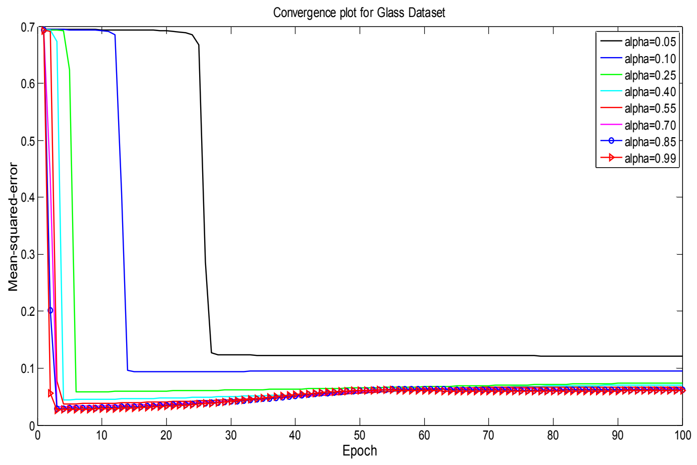


Figure. 2.4. Convergence plot obtained by employing fuzzy-MLP algorithm for Glass dataset

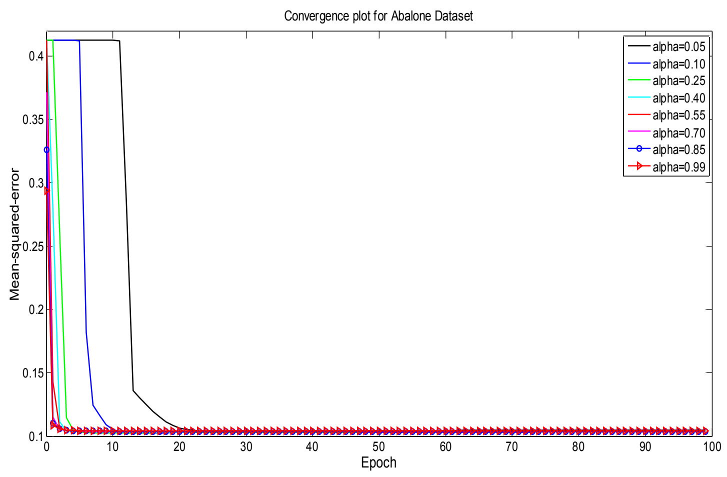


Figure. 2.5. Convergence plot obtained by employing fuzzy-MLP algorithm for Abalone dataset

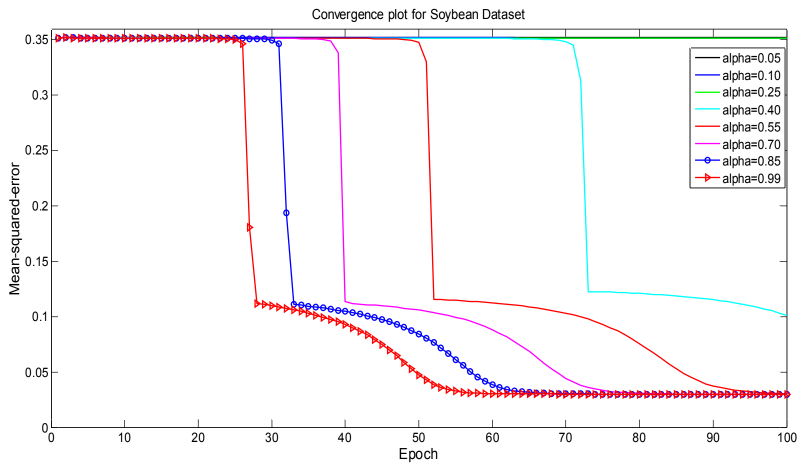


Figure. 2.6. Convergence plot obtained by employing fuzzy-MLP algorithm for Soybean dataset

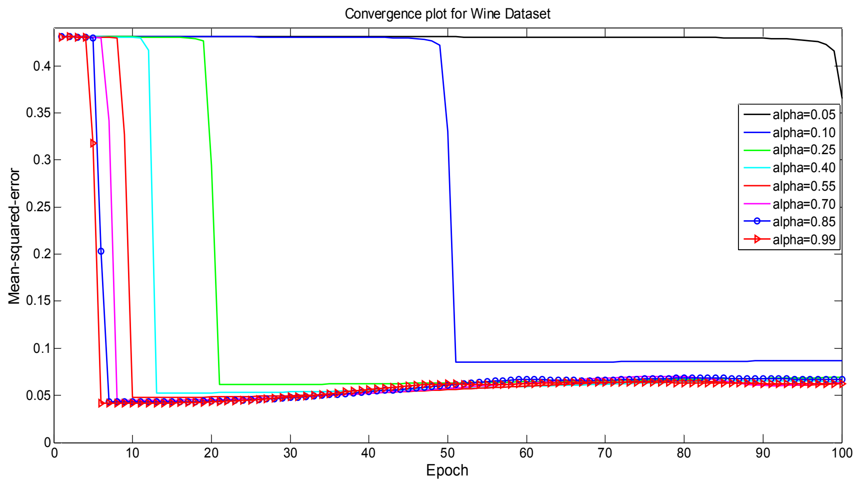


Figure. 2.7. Convergence plot obtained by employing fuzzy-MLP algorithm for Wine dataset

For all the datasets, the proposed algorithm results a higher gain when compared with the MLP model. However, as the MSE varies with α value, it will be interesting to analyze a plot for the gain against each α. A plot has been given in Figure 2.8 by α-value in X-axis and average gain (average C_g) in Y-axis. It should be noted that average gain against an α is obtained by calculating the average of all the gains obtained with the six datasets used in this work. Figure 2.8 clearly reveals that best gain, which is close to 95%, is obtained when the parameter α is tuned to 0.55, 0.85 and 0.99. So, it should be noted that for any datasets including the six UCI datasets used in this work, the proposed NN model will achieve a higher convergence gain. However, it will also be important to check if the simulation time is affected by α. Therefore, a plot has been given in Figure 2.9 to show a comparative characteristic of average simulation time for different α values. As mentioned in previous section, the simulation time is obtained by calculating mean of simulation time obtained with five consecutive executions to make the result error free. In this plot, α has been taken in X-axis and average simulation time is shown in Y-axis.

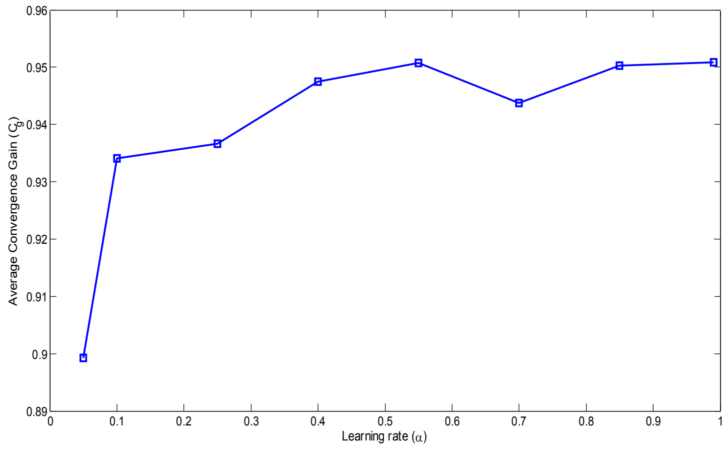


Figure 2.8. Average gain obtained against different learning rate (α)

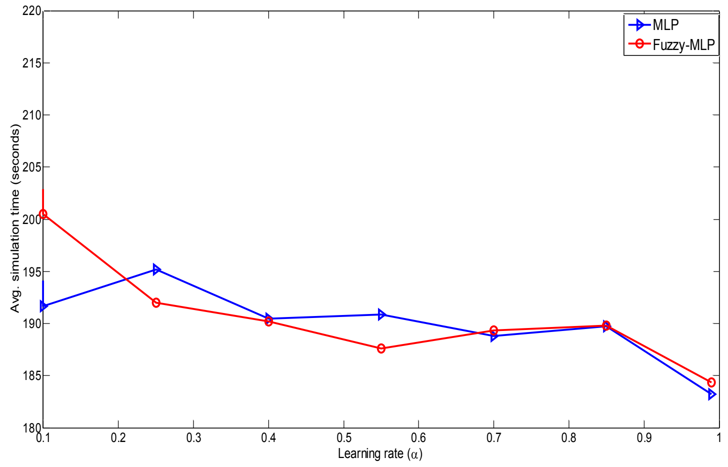


Figure 2.9. Average simulation time obtained against different learning rate (α)

In the previous discussion, it is seen that when α is tuned to 0.55, 0.85 or 0.99, the proposed model obtains a maximum of 95% convergence gain. So, it will be wise to check the simulation time against these three α values only. Figure 2.9 shows, for α=*0.55,* the proposed model is trained within 190 seconds whereas the MLP model is taking more than 190 seconds for the purpose. For rest two α values, the simulation is approximately equal to that obtained by using the MLP model and it is more than that obtained by setting α to 0.55. Therefore, it could be concluded that the best α is 0.55 for all the future tests.

2.4. Conclusion

The fuzzy-MLP for pattern classification has been developed. The input patterns are fuzzified by using spline (S-shaped) membership function and then input to the MLP model. The results obtained shows that, the proposed model converges to its minimum MSE within 100 epochs and achieves a convergence gain of 93%. The proposed algorithm outperforms MLP for all the six UCI datasets used in this work. As future work, it will be advantageous to optimize the network with an optimization algorithm.

References

[1] G.P. Zhang, "Neural Networks for Classication: A Survey," in IEEE Transactions on Systems, Man,and Cybernetics Part C: Applications and Reviews, Vol. 30, No. 4, November 2000, pp. 451-462.

[2] L. Liu, X. Peng, "An Improved Diagonal Recurrent Neural Network and Its Application in Nonlinear Time Series Forecasting," in Information Engineering Letters, ISSN: 2160-4114 Volume 2, Number 1, March, 2012, pp. 18-25.

[3] W. Sibanda, P. Pretorious, "Novel Application of Multi-Layer Perceptrons (MLP) Neural Networks to Model HIV in South Africa using Seroprevalence Data from Antenatal Clinics," in International Journal of Computer Applications, Volume 35:5, December 2011, pp. 26-31.

[4] P.K. Dash, H.S. Behera, I.W.C. Lee, "Time sequence data mining using time-frequency analysis and soft computing techniques," Applied Soft Computing, 8(1):202-215 (2008).

[5] H.S. Behera, P.K. Dash, B.N. Biswal, "Power quality time series data mining using S-transform and fuzzy expert system," Applied Soft Computing, 10(3): 945-955 (2010).

[6] C. L. Liu, F. Yin, D. H. Wang, Q. F. Wang, "Online and oine handwritten Chinese character recog-nition: benchmarking on new databases," in Pattern Recognition, 46(1), 155-162, 2013.

[7] R. Kumar, J. D. Sharma, B. Chanda, "Writer-independent online signature verification using surroundedness feature," in Pattern Recognition Letters, 33(3), 301-308, 2012.

[8] T. Dash, T. Nayak, S. Chattopadhyay, F. A. Rabhi, "A Fingerprint Verification Tool using Adap-tive Resonance Theory Nets," in American Journal of Biomedical Engineering, 3(2):31-40; doi:10.5923/j.ajbe.20130302.01.

[9] C. Kurian, K. Balakrishnan, "Continuous speech recognition system for Malayalam language using PLP cepstral coefficient," in Journal of Computing and Business Research, Vol.3, Issue.1, 2012.

[10] S. Chattopadhyay, "Neurofuzzy models to automate the grading of old - age depression," in Expert Systems, doi: 10.1111/exsy.12000, 2012.

[11] M. Kang, D.P. Brown, "A modern learning adaptive functional neural network applied to handwritten digit recognition," in Information Sciences, 178(2008), 38023812.

[12] S. Mitra, R. K. De, S. K. Pal, “Knowledge-Based Fuzzy MLP for Classification and Rule Generation,” in IEEE Transactions on Neural Networks, Vol. 8, No. 6, November 1997, pp. 1338-1350.

[13] S.N. Sivanandan, S.N. Deepa, “Principle of Soft Computing (2nd Edition),” WILEY-INDIA, ISBN -978- 81- 265 - 2741- 0, 2011.

[14] K. Bache, M. Lichman, “UCI Machine Learning Repository [http://archive.ics.uci.edu/ml] Irvine, CA,” University of California, School of Information and Computer Science, 2013.

Chapter-3

3

A Metaheuristic Fuzzy MLP Approach for Non-linear Classification and its Experimental Analysis

Preview:

This chapter presents three metaheuristics based training algorithms for Fuzzy MLP. The developed models are Genetic algorithm (GA) based Fuzzy MLP, Particle Swarm Optimization (PSO) based Fuzzy MLP and Gravitational search technique (GS) based Fuzzy MLP. Comparative performance analysis has been carried out on the developed models.

3.1. Motivation

Pattern Classification is the organization of patterns into different groups sharing similar property(s). Pattern classification has successfully been applied many areas of science and engineering in the literature that includes business, science, industry, and medicine etc. A few examples are financial time series forecasting [3,4], financial data classification [3,4], medical diagnosis [5,6], time sequence data mining [7], power quality control and classification [8,9], handwritten character recognition [10,11,12], signature verification [13,14], fingerprint recognition [15], speech recognition [16] etc.

Therefore, in this work, an attempt has been made to propose use of certain optimization algorithm for training Fuzzy MLP for pattern classification problems.

3.2. Related works

Research has been carried out in the field of pattern classification using statistical techniques or by employing data mining techniques, such as neural computing. Statistical methods use probability theory, decision theory to classify a set of data [17]; whereas in neural computing techniques, Neural Networks (NNs) are used for pattern classification [18]. With increasing complexities of real world classification problems, the NN technique is gaining popularity which in turn is increasing the field of development of new NN models, new training algorithms. The statistical methods use certain set of assumptions and conditions to satisfy the solution. And these developed models cannot be applied directly to any such problems [19]. However the first and the foremost advantage of using NN techniques is that, NNs are nonlinear models which make them adjustable to the real world input data without any explicit specification or external conditions. Secondly, they are universal functional approximation models which can approximate any function with arbitrary accuracy [19,20,21].

Significant progress has been made in recent years in the area of pattern classification using neuro-computing techniques. Feed-forward Multilayer Perceptrons (MLPs) are the most widely used NN classifiers. The architecture of MLP contains more than two layers which are (i) input layer, (ii) output layer and (iii) hidden layer (s). Depending on application, the number of hidden layer can be increased or decreased. However, Kang and Brown [22] developed an unsupervised NN for pattern classification, called as adaptive function neural network which has no hidden layer in its architecture. Limitation of MLP is that it can barely handle uncertainty and impreciseness of input and output data. The back propagation training algorithms, which are based on gradient descent, incrementally reduce the output error. Although this algorithm is effective for wide range of classification problems, they suffer from two significant drawbacks. First, they are restricted to find the local minima. Second, they get stuck in the flat region of the search space, where the algorithm needs to restart with a new set of inputs. Therefore, in this work, back propagation algorithm has not been used; instead GA, PSO or GS algorithms are used for training the developed fuzzy MLP.

3.2.1. Genetic algorithm (GA)

GA is population based algorithm inspired by the theory of biological evolution [23]. An individual in GA is termed as a chromosome. Based on encoding of chromosomes, there are two types of GA viz. binary coded GA (BCGA) or real coded GA (RGA). In BCGA, each chromosome is encoded with a series of 0s or 1s. But, in RGA each chromosomes are encoded with real numbers. In this paper, RGA has been used to train the developed NN for automating the process of setting initial parameters such as weights, learning rate, and momentum factor. GA chooses an initial population of chromosomes at random and generates a new set of chromosomes by performing crossover and mutation among them until best offspring is obtained.

3.2.2. Particle swarm optimization (PSO)

PSO is also a population based stochastic optimization technique developed by Eberhart and Kennedym [24]. PSO shares many similarities with GA. In both the algorithm, the system is initialized with a set of random population and searches for optimum by updating generations. PSO starts with the random initialization of a population (swarm) of individuals (particles) in the *n*-dimensional search space. The particles fly over search space with adjusted velocities. [25] In PSO, each particle keeps two values in its memory: (1) its own best experience, that is, the one with the best fitness value (best fitness value corresponds to least objective value since fitness function is conversely proportional to objective function), whose position and objective value are called P_i and P_{best}, respectively, and (2) the best experience of the whole swarm, whose position and objective value are called P_g and g_{best}, respectively [29]. Let denote the position and velocity of particle i with the following vectors: $X_i = (X_{i1}, X_{i2}, X_{i3} \ldots X_{in})$ and $V_i = (V_{i1}, V_{i2}, V_{i3} \ldots V_{in})$

The updated velocities and positions of the particles can be calculated according to the following equations:

$$V_{i+1} = V_i + c_1\theta_1 \times (pBest - P_i) + c_2\theta_2 \times (gBest - X_i) \qquad (1)$$

$$P_{i+1} = P_i + \Delta \times V_i \qquad (2)$$

Where c_1 and c_2 are two positive numbers, and θ_1 and θ_2 are two random numbers with uniform distribution in the interval [0,1].

3.2.3. Gravitational Search (GS)

GS algorithm is based on the law of gravity or law of masses [26,27]. In GS, agents are considered as objects and their performance is measured by their masses. According to law of gravity, each object attracts every other object by the gravitational force, and objects with lighter masses are attracted towards heavier masses, causing a global movement of all the objects present in the space. The solution is exploited by movement of heavier masses, which move more slowly than lighter ones [27]. In this process a solution is called as a fitness value.

After finding a fitness value, the mass of each object is calculated as given in Equation-3 below.

$$M_i = \frac{fit_i - worst}{\sum_{j=1}^{n}(fit_j - worst)} \tag{3}$$

Where *fit$_i$* is the fitness value of agent i, in the current epoch evaluation and *worst* is defined as given in Equation-4 below.

$$worst = \max_{j\in\{1,\ldots,N\}} fit_j \tag{4}$$

The force acting on object *i* from object *j* is defined as given in equation-5.1 below. Similarly, summing up all the individual forces acting on particle *i,* we will get the total force as given in equation-5.2.

$$Fij = G_0 \frac{M_i \times M_j}{R_{ij} + \theta}(x_j - x_i) \tag{5.1}$$

$$\begin{aligned} F_i &= \sum_{j=1, j\neq i}^{n} r_j Fij \\ \Rightarrow F_i &= G_0 M_i \sum_{j=1, j\neq i}^{n} \frac{Mj}{R_{ij} + \theta}(x_j - x_i) \end{aligned} \tag{5.2}$$

Acceleration of object *i* is calculated using equation-6.

$$a_i = F_{ij} / m_i \tag{6}$$

The velocity and next position of the object *i* will be calculated using equation-7 and 8 respectively as below.

$$V_{next} = V_{current} \times r_i + a_i \tag{7}$$

$$x_{next} = x_{current} + V_{new} \tag{8}$$

Where R_{ij} is defined as the Euclidean distance between two objects *i* and *j*, and is given in equation 11. r_1, r_2 and θ are arbitrary random values.

This paper proposes a development of GA, PSO and GS based Fuzzy MLP for real world pattern classification. The resulting issues are being discussed in later sections. It should be noted that the algorithm can also be employed to other NN models like Hopfield network, higher order neural networks, Adaptive Resonance Theory (ART), Support Vector Machine (SVM) etc.

3.3. Fuzzy-MLP for Pattern Classification

The proposed fuzzy MLP in this work uses an S-shaped (S-shaped or spline) membership function (MF) to fuzzify the input dataset. It should be noted that spline MF is considered very simple which does not used any exponential terms. This makes the fuzzified data accurate and precise. And it normalizes the input value to a certain range. Figure 3.1 represents a spline MF where *a*, *b* locate the extremes of sloped portion of the curve. Equation 9 describes a spline MF which can be used to fuzzify the value *x*.

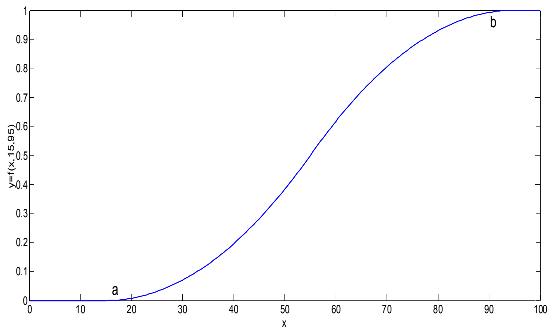


Figure 3.1 S-shaped MF

$$f(x,a,b)=\begin{cases} 0, x<a \\ 2\left(\dfrac{x-a}{b-a}\right)^2, a\le x\le \dfrac{a+b}{2} \\ 1-2\left(\dfrac{x-b}{b-a}\right)^2, \dfrac{a+b}{2}\le x\le b \\ 1, x>b \end{cases} \quad (9)$$

3.3.1. Fuzzy-MLP network architecture

The proposed architecture is similar to that of the classical MLP architecture and has been shown in Figure 3.2 below.

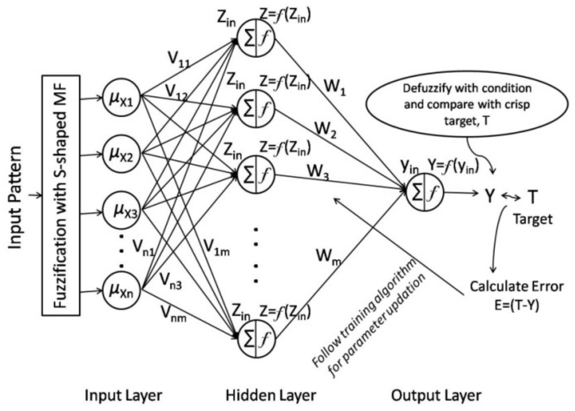


Figure 3.2 Fuzzy MLP architecture

3.3.2. Fuzzy-MLP learning algorithm

```
Initialize Learning rate (α), momentum factor (μ)
Initialize W, V, $b_1$, $b_2$ using GA/PSO/GS
Fuzzify the training input pattern using S-shaped MF
Input the fuzzified values to the input layer neurons
Set mean-squared-error, MSE = 0;
While (termination criteria is not satisfied)
```

```
Do
    For each training pattern
        For each hidden neuron
            Calculate input to the hidden neuron ($Z_{in}$) using equation-11
            Calculate activation of hidden neuron, $Z = f(Z_{in})$
        End for
        Calculate net input to the output neuron ($y_{in}$) using equation-12
        Calculate activation of output neuron, $Y = f(y_{in})$
        Y=Defuzzify the fuzzy output Y ($\mu_Y$) with threshold condition
        Compute the error, $E = T–Y$; where T is the corresponding target
        Update the weights (W, V, b1, b2) using GA/PSO/GS
    END FOR
    Update $MSE = (MSE + E^2)/n$
End while
```

It should be noted that all the three training algorithms (GA, PSO and GS) are based on updation of weights (Hidden-Output layer weight: ***W***, Input-Hidden layer weight: ***V***, Biases in hidden layer: ***b₁***, Bias in output layer: ***b₂***). The activation function used in hidden as well as in the output layer is the binary sigmoid, which is defined by following formula:

$$f(x) = \frac{1}{1+e^{-x}} \tag{10}$$

$$Zin_j = v_{ij} + b_j + \sum_{i=1}^{n} x_i v_{ij} \tag{11}$$

$$y_{in} = w_i + b + \sum_{i=1}^{NumberofHiddenUnits+1} z_i w_i \tag{12}$$

Mean squared error (MSE) can be computed using the following equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - t_i)^2 \tag{13}$$

Where x_i and t_i are computed value and target value respectively for *i*-th instance of the dataset.

3.3.3. GA based training algorithm

```
Start
Generate n chromosomes at random
Initialize mutation probability ($P_m$)
Initialize W, V, $b_1$, $b_2$ with the randomly generated chromosomes
While termination condition is not achieved
    Calculate fitness [Mean Squared Error (MSE)] for each chromosome for each epoch
```

```
        Sort the population in ascending order of their fitness (MSE) i.e. minimum to
        maximum MSE
        Replace the lowest fit chromosomes with highest fit chromosomes
        Perform crossover
        Mutate the chromosomes with mutation probability ($P_m$) at each locus (position)
        Place new child chromosomes in the population
        Use the new population for further run of the algorithm
End while
Test the trained Fuzzy MLP for finding pattern classification accuracy
Stop
```

3.3.4. PSO based training algorithm

```
Start
Initialize particle dimension, no. of particles, inertia weight (w), maximum and minimum
inertia weight ($w_{max}$, $w_{min}$), coefficients ($c_1$, $c_2$), delta ($\Delta$)
Initialize velocity (V), position (P) of each particle, local and global best score (pBestScore,
gBestScore), gBest to 0
While termination condition is not achieved
        For each particle
                Calculate activation of Fuzzy MLP
                Calculate average fitness
                If the fitness is better than the previous, set the current pBestScore=fitness
                According to iii, set the best position of the particle
                Calculate the best fitness for neighbor particles (gBestScore)
                Update inertial weight using Equation-2.
                Update velocity and position of particle using Equation-1 and 2 respectively
        End for
End while
Test the trained Fuzzy MLP for finding pattern classification accuracy
Stop
```

3.3.5. GS based training algorithm:

```
Start
Initialize n-dimensional swarm of objects, gravitational constant ($G_0$), velocity, acceleration,
masses of objects, force
While termination condition is not achieved
        For each object i=1 to n
                Calculate activation of Fuzzy MLP
                *Evaluate the fitness (MSE)*
                *Calculate mass using Equation-3*
                *Calculate acceleration using Equation-6*
                *Update the velocity using Equation-7*
                *Update position of object using Equation-8*
        **End for**
**End while**
*Test the trained Fuzzy MLP for finding pattern classification accuracy.*
**Stop**
```

3.4. Experimental Results

The MLP and Fuzzy MLP algorithms are implemented using MATLAB R2010a installed in a PC with Windows OS and 2GB of main memory. The processor is Intel dual processor system (Intel Core2Duo) and each processor has equal computation speed of 2 GHz. Seven datasets such as one bit parity matrix of 4 bit XOR operation and six UCI datasets *(http://archive.ics.uci.edu/ ml/)* are tested against each algorithm for this work. To ensure that the input values were compatible despite significant differences in their values, the dataset is normalized with respect to each input value using the following formula:

$$X = \frac{X_i - X_{i\min}}{X_{i\max} - X_{i\min}}; i = 1,2,...n \tag{14}$$

3.4.1. Description of datasets

All the seven datasets considered for this work has been briefly described below and the Table 3.1 shows a summary of the main characteristics of the following datasets.

4-bit XOR data (XOR4): A total of 16 instances of 4 bit decimal XOR parity data are generated for initial implementation of the six models (MLP and Fuzzy MLP along with three training algorithms). Here there are two classes based on whether the output is 0 or 1.

Iris dataset: This is a popular dataset based on multivariate characteristics of flower plant species. Those are length and thickness of petals and sepals. The dataset contains three classes such as Iris Setosa, Iris Versicolor and Iris Virginica of 50 instances each. The dataset contains 150 instances and 5 attributes (4 predicting and 1 target). All the attribute values are real.

Bank note authentication (BNA) dataset: This dataset is a collection of 5 attributes of a bank note. Based on these dataset, the note can be original or duplicate. A total of 1372 instances are presented in the dataset.

Blood transfusion (BT) dataset: The dataset was taken from the blood transfusion service center in Hsin-Chu city in Taiwan and contains multivariate attributes. Number of instances are 748. The attributes are recency, frequency, monetary, time and a target attribute representing whether the donor has donated blood in previously in March 2007.

Hayes-Roth (HR) dataset: It is a dataset prepared for psychological research and contains attributes like names, hobby, age, educational level, marital status and the class of the person. However, the values are converted to real by the donor of the dataset.

Teaching assistant evaluation (TAE) dataset: The dataset consists of evaluations of teaching performance of 151 TAs in 5 semesters. The scores were grouped into three classes: low, medium and high to form the class variable.

User knowledge modeling (UKM) dataset: This dataset is based on development of software products. The attributes contained in the dataset are: the degree of study time for goal object materials, the degree of repetition number of user for goal object materials, the degree of study time of user for related objects with goal objects, the exam performance of user for related objects with goal object, the exam performance of user for goal objects and the knowledge level of user, which can be very low to high.

Table 3.1 Dataset properties

Dataset	No. of Patterns	No. of Training patterns	No. of Testing patterns	No. of target classes
XOR4	16	11	5	2
Iris	150	105	45	3
BNA	1372	960	412	2
BT	748	523	225	2
HR	160	112	48	3
TAE	151	105	46	3
UKM	403	282	121	4

3.4.2. Statistical analysis of dataset

A statistical analysis of the data has been carried out to observe distribution of attribute values within the dataset. Three statistical measures have been computed for this purpose which is given below. The values for each dataset are included in Tables 3.2(a)–3.2(f).

- *Mean (M):* Average or mean value of array
- *Standard deviation (±SD):* shows how much variation or dispersion from the mean exists for the attribute.
- *Skewness (SK): SK* is the measures of symmetrical distribution of the dataset over the mean. As the XOR4 matrix is small, the analysis is not necessary for this case. The skewness values which are near to 0 are symmetrically distributed in both positive and negative region of axis. If the SK value is negative, then the attribute is distributed more towards left and if the SK value is positive, the attribute is distributed more towards right.

Table 3.2(a). Statistical results of Iris dataset

Attributes	A1	A2	A3	A4
M	5.8433	3.0540	3.0540	1.1987
±SD	0.8281	0.4336	1.7644	0.7632
SK	0.3118	0.3307	-0.2717	-0.1039

Table 3.2(b). Stat. results of BNA dataset

Attributes	A1	A2	A3	A4
M	0.4333	1.9224	1.3976	-1.1917
±SD	2.4828	5.8690	4.3100	2.1010
SK	-0.1492	-0.3937	1.0874	-1.0211

Table 3.2(c). Statistical results of BT dataset

Attributes	A1	A2	A3	A4
M	9.5067	5.5147	1378.7	34.2821
±SD	8.0954	5.8393	1459.8	24.3767
SK	1.8767	3.2048	3.2048	0.7479

Table 3.2(d). Statistical results of HR dataset

Attributes	A1	A2	A3	A4	A5
M	66.500	2.00	1.95	1.95	1.95
±SD	38.24	0.81	0.94	0.94	0.94
SK	0	0	0.73	0.73	0.73

Table 3.2(e). Statistical results of TAE dataset

Attributes	A1	A2	A3	A4	A5
M	1.80	13.64	8.10	1.84	27.86
±SD	0.39	6.82	7.02	0.36	12.89
SK	-1.56	-0.008	0.86	-1.93	0.49

Table 3.2(f). Stat. results of UKM dataset

Attributes	A1	A2	A3	A4	A5
M	0.37	0.35	0.46	0.43	0.45
±SD	0.21	0.21	0.24	0.24	0.25
SK	0.54	0.60	-0.05	0.39	0.07

3.4.3. Classifier training results

To classify patterns with MLP and Fuzzy MLP using GA, following settings used. Number of population is set to 50, probability of crossover (P_c) is set to 0.5, mutation probability (P_m) is set to 0.3 [28]. Similarly, the parameter setting is as follows. Number of particles is set to 50. Inertia weight (w) is set to 2. w_{max} and w_{min} are set to 0.9 and 0.5 respectively, as these two values are optimal for classification using PSO [29]. The coefficients, c_1 and c_2 are set to 2 each. The GS parameters are: population size is 50, object dimension is 30, and gravitational constant (G_0) is set to 1 [26,27].

It should be noted that all the algorithms are run on the datasets for 100 epochs and corresponding result was noted against each hidden node setting (nh). The detailed result for MLP and Fuzzy MLP for all the three developed training algorithms are given in Appendix-1. Based on these values, a few observations are being made from the plots as shown in Figures 3.3–3.10. But, as the work focuses on the development of Fuzzy MLP, the plots are being generated by taking epoch in X-axis and MSE in Y-axis.

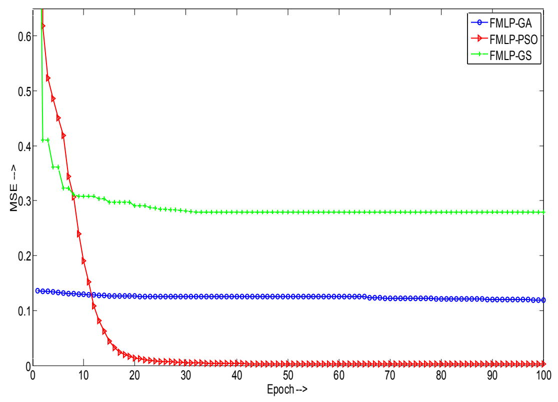


Figure 3.3 Training with XOR4 and 4 hidden nodes

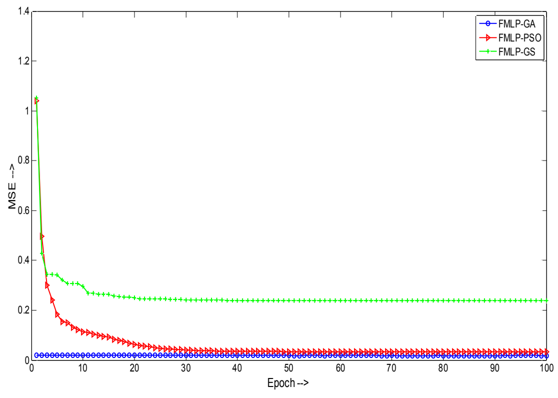


Figure 3.4 Training with Iris and 8 hidden nodes

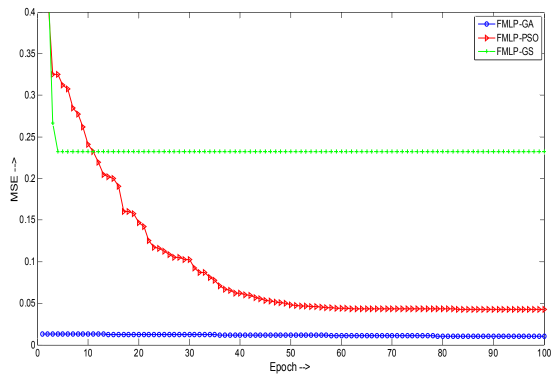


Figure 3.5 Training with BNA and 20 hidden nodes

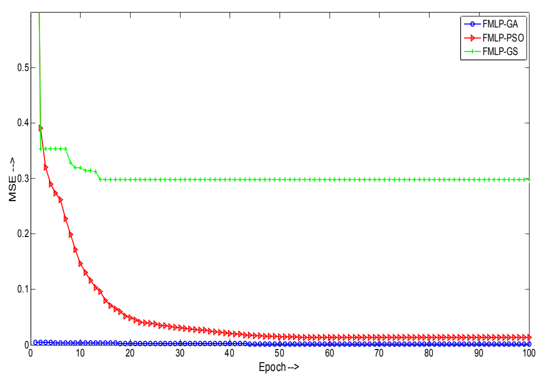


Figure 3.6 Training with BT and 16 hidden nodes

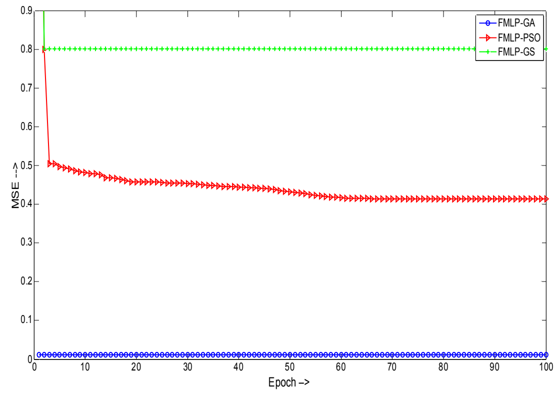


Figure 3.7 Training with HR and 20 hidden nodes

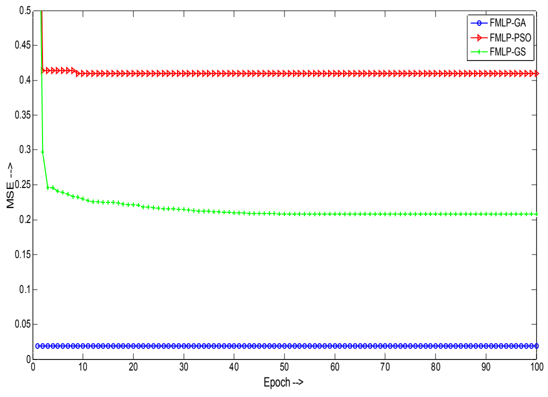


Figure 3.8 Training with TAE 12 hidden nodes

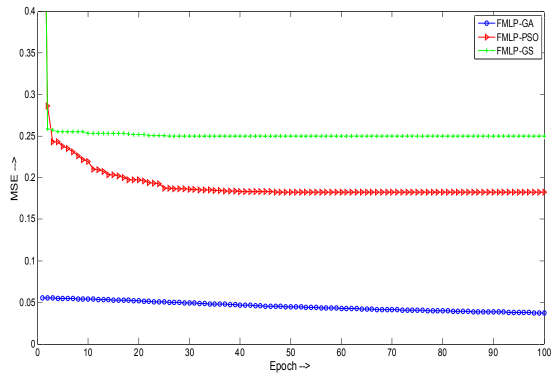


Figure 3.9 Training with UKM, 12 hidden nodes

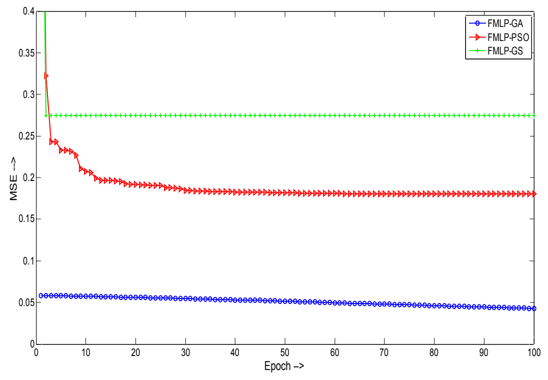


Figure 3.10 Training with UKM, 16 hidden nodes

3.4.4. Testing results

After carefully training the classifiers with three training algorithms, the testing process has been followed. Major difference between training and testing is that, training process updates the knowledge of the NNs whereas the testing uses the updated knowledge. The results of the testing process are noted against each hidden layer setting along with the classification time in seconds. The detailed results are given in Tables 3.3(a)–3.3(g) below. It should be noted that while calculating classification accuracy (using $N\times100/M$ %, where N is the number of data correctly classified and M is the total number of data present in the dataset), the output error is compared against a threshold value. This threshold value is obtained from the number of target classes and the minimum target value. If the error is less than or equal to this threshold, then the pattern is recognized and assigned the corresponding class level else it is not recognized by the classifier. The classification accuracy is sometimes very close to 100 for some datasets and algorithms, so the precision is removed and represented as ~100 in the Tables.

In Tables 3.3(a)–3.3(g), some data are bold faced showing the best result obtained with the classifiers with various hidden layer setting. Table 3.3(a) shows the Fuzzy MLP trained with PSO algorithm is giving maximum accuracy with very less testing time for all the values of *nh*. For Iris dataset, the proposed Fuzzy MLP with GA and PSO is giving the best accuracy. The maximum accuracy is very close to 100% (see Table 3.3(b)). But, when the bank note authentication dataset is tested for all the classifiers, GS based Fuzzy MLP is showing better accuracy than GA and PSO. But, again when the numbers of hidden nodes is increased from 16 to 20, the performance of the same is degrading and maximum time is consumed. However, with this number of hidden layers, one can easily choose PSO, as it is giving a 97% accuracy for the mentioned dataset (refer Table 3.3(c)). Blood transfusion dataset is successfully tested with GA and PSO based Fuzzy MLP with a maximum accuracy of 99.9%. Hayes-Roth and Teaching assistant evaluation datasets have been successfully tested with GA based Fuzzy MLP for the entire hidden layer setting with accuracies of 97.7% and 93.4% respectively (refer Table 3.3(e) and 3.3(f)). But while the User knowledge modeling dataset is tested for the developed classifiers, it is coming up with an approximate maximum accuracy of 70%. Table 3.4 shows average accuracy for all the datasets corresponding to each hidden node setting and a related plot has been shown in Figure 3.11 by taking number of hidden nodes in X-axis and average classification accuracy in Y-axis. The Table shows that fuzzy MLP when trained with GA is providing better results as compared to MLP or fuzzy MLP trained with PSO or GS. The maximum average accuracy could be obtained by setting number of hidden nodes to 12 for GA based fuzzy MLP and is found to be 82.57%.

The testing time consumed by all the algorithms is observed to be increasing with number of hidden units [30]. Taking this into account, average testing time consumed by each classifier is presented in Table 3.5. A plot has also been given in Figure 3.12 to compare MLP and Fuzzy MLP with respect to average testing time they consume. Both GA based MLP and Fuzzy MLP are consuming approximately same time for decision, where as PSO and GS based Fuzzy MLP are producing their outputs with more than the time taken by the MLP. The plot given in Figure 3.12 also reveals that, the hidden layer plays an important role in deciding the efficiency of classifiers and the classification time.

3.4.5. Statistical analysis of test results

ANOVA test on the results has been conducted SPSS software package [31] carefully to see if there is any difference between the groups on the resulted data. Simple t-test results using

ANOVA for all the three versions Fuzzy MLP have been given in Tables 3.6(a)–3.6(c). The Table gives of t-value, mean difference and relative mean for each dataset.

Table 3.3(a). Classification result for XOR4 dataset

Classifier ↓		Number of hidden nodes (*nh*)									
		nh=4		*nh*=8		*nh*=12		*nh*=16		*nh*=20	
		Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)
GA	MLP	50	1.17	25	5.19	25	32.3	25	67.5	25	119
	Fuzzy MLP	50	0.98	50	5.14	50	30.1	50	67.3	50	122
PSO	MLP	56.2	2.57	87.5	1.65	62.5	1.99	81.25	2.31	50	2.52
	Fuzzy MLP	~100	3.13	~100	3.46	~100	4.05	~100	4.27	~100	4.63
GS	MLP	31.25	2.60	25	3.89	43.7	5.58	31.25	6.81	56.2	8.29
	Fuzzy MLP	~100	4.42	~100	5.81	~100	7.34	50	8.80	50	10.2

Table 3.3(b). Classification result for Iris dataset

Classifier ↓		Number of hidden nodes (*nh*)									
		nh=4		*nh*=8		*nh*=12		*nh*=16		*nh*=20	
		Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)
GA	MLP	65.3	3.32	78.67	9.90	82.67	17.9	82.67	28.3	82.67	40.8
	Fuzzy MLP	96.6	3.92	96.7	9.98	97.3	18.4	97.3	28.5	97.3	41
PSO	MLP	81.3	8.82	84.67	12.0	94	14.9	94	16.4	91.3	18.9
	Fuzzy MLP	~100	27.4	93.3	32.5	96	32.5	96	35.1	90.67	37.8
GS	MLP	54.0	10.5	59.3	14.1	64.67	17.9	56.0	21.9	33.33	25.4
	Fuzzy MLP	35.33	29.5	78.67	32.9	34.0	36.4	83.33	40.9	50.0	42.8

Table 3.3(c). Classification result for Bank note authentication dataset

Classifier ↓		Number of hidden nodes (*nh*)									
		nh=4		*nh*=8		*nh*=12		*nh*=16		*nh*=20	
		Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)
GA	MLP	74.70	25.8	74.78	88.7	74.63	162	74.41	257	74.34	369
	Fuzzy MLP	88.55	28.6	89.79	88.2	90.08	163	88.33	258	85.78	372
PSO	MLP	96.9	78.9	95.6	102	87.3	124	93.7	154	97.5	182
	Fuzzy MLP	96.1	289	93.7	312	90.5	339	97	369	88	398
GS	MLP	45	78	57.4	103	88.3	128	35	153	47	178
	Fuzzy MLP	92.3	282	74.7	309	90	334	40.1	358	69.9	388

Table 3.3(d). Classification result for Blood transfusion dataset

Classifier ↓		Number of hidden nodes (*nh*)									
		nh=4		*nh*=8		*nh*=12		*nh*=16		*nh*=20	
		Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)
GA	MLP	78.7	14	89.6	49	85.1	89	86.5	141	74.9	203
	Fuzzy MLP	99.9	15.4	99.9	48.4	99.9	88.7	99.9	140	99.9	204
PSO	MLP	78.5	44	77.8	57	77.1	72	78.2	91	76.6	104
	Fuzzy MLP	86	150	86.2	164	72.3	175	89.4	199	91.4	219
GS	MLP	76.2	44	76.2	61	76.2	74	75.7	90	76.2	105
	Fuzzy MLP	64.6	151	50.2	165	64.6	179	56.1	195	56.7	210

Table 3.3(e). Classification result for Hayes-Roth dataset

Classifier ↓		Number of hidden nodes (*nh*)									
		nh=4		*nh*=8		*nh*=12		*nh*=16		*nh*=20	
		Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)
GA	MLP	58.3	2.7	58.3	9.3	58.3	16.8	58.3	27.1	58.3	41.6
	Fuzzy MLP	97.7	2.61	93.4	9.14	94.7	17.3	97	27.7	92	40.0
PSO	MLP	37.9	8.2	40.2	11	47.7	13.8	38.6	16	50	18.5
	Fuzzy MLP	62.9	26.2	53	28.6	65.9	31.3	66.7	33.8	59.1	37.1
GS	MLP	45	9.7	37.8	12.8	24.2	16.3	22.7	20.1	15.2	23.5
	Fuzzy MLP	75.8	26	75	29	67.9	33	71.1	37	50.8	40

Table 3.3(f). Classification result for Teaching assistant evaluation dataset

Classifier ↓		Number of hidden nodes (*nh*)									
		nh=4		*nh*=8		*nh*=12		*nh*=16		*nh*=20	
		Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)
GA	MLP	55.6	3	99.3	11	~100	19	~100	30	~100	43
	Fuzzy MLP	93.4	4.6	93.4	10.8	93.4	19.3	93.4	30.2	93.4	44.8
PSO	MLP	37.9	8.2	40.2	11	47.7	13.8	38.6	16	50	18.5
	Fuzzy MLP	62.88	26	53.03	29	65.91	31	66.67	34	59.09	37
GS	MLP	23	10.4	19.8	14.2	29.1	18.2	26.4	22.1	17.2	26.2
	Fuzzy MLP	19.21	29	67.9	33	33.77	37	56.3	42	54.3	45

Table 3.3(g). Classification result for User knowledge modeling dataset

Classifier ↓		Number of hidden nodes (*nh*)									
		nh=4		*nh*=8		*nh*=12		*nh*=16		*nh*=20	
		Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)	Accuracy (%)	Time (sec.)
GA	MLP	10	5.1	10	17.3	27.3	31.7	32	50.1	31.6	72.8
	Fuzzy MLP	50	4.9	51.3	17.9	52.6	32	50.8	50.2	50.8	74.2
PSO	MLP	24.42	15.7	25.58	20.6	25.19	25.4	25.97	30.3	24.81	35
	Fuzzy MLP	63.57	64.9	64.34	54.5	63.57	59.7	62.79	64.4	61.63	69.3
GS	MLP	24.42	16.7	24.42	22.4	16.67	28.1	23.64	33.9	10.8	39.9
	Fuzzy MLP	51.6	49.5	56.9	55.3	68.8	61.4	60	67.1	58	73.2

Table 3.4 Comparison of MLP and Fuzzy MLP based on average accuracy

Classifier ↓		Average accuracy (%)				
		nh=4	*nh*=8	*nh*=12	*nh*=16	*nh*=20
GA	MLP	56.086	62.236	58.833	59.813	57.802
	Fuzzy MLP	82.307	82.070	82.569	82.390	81.311
PSO	MLP	59.017	64.507	63.070	64.331	62.887
	Fuzzy MLP	74.290	73.928	75.697	79.760	74.982
GS	MLP	42.696	42.846	48.977	38.670	36.561
	Fuzzy MLP	56.473	67.228	59.845	59.561	55.671

Table 3.5 Comparison of MLP and Fuzzy MLP based on average testing time

Classifier ↓		Average testing time (sec.)				
		nh=4	*nh*=8	*nh*=12	*nh*=16	*nh*=20
GA	MLP	7.870	27.199	52.671	85.857	127.029
	Fuzzy MLP	8.716	27.080	52.686	85.986	128.286
PSO	MLP	23.770	30.750	37.984	46.573	54.203
	Fuzzy MLP	83.804	89.151	96.079	105.653	114.690
GS	MLP	24.557	33.056	41.154	49.687	58.041
	Fuzzy MLP	81.631	90.001	98.306	106.971	115.600

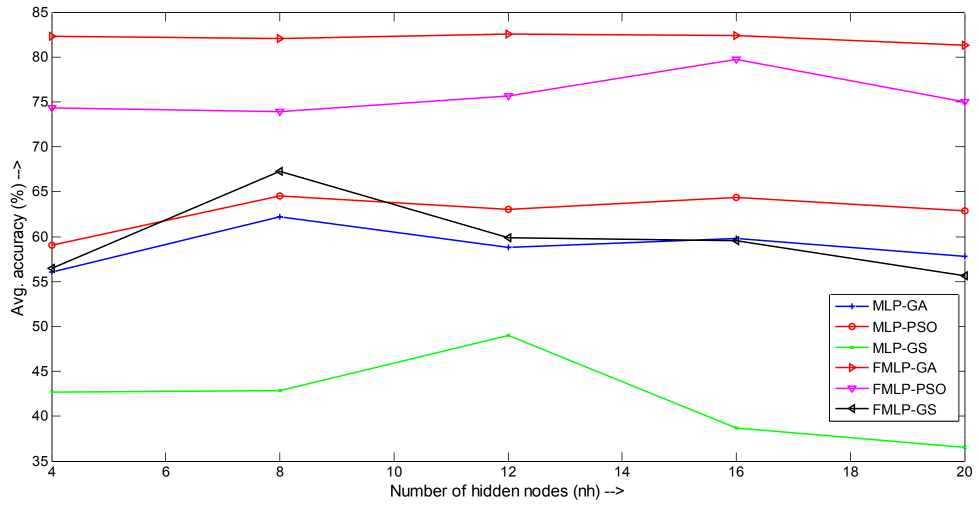


Figure 3.11 Effect of number of hidden nodes on average accuracy

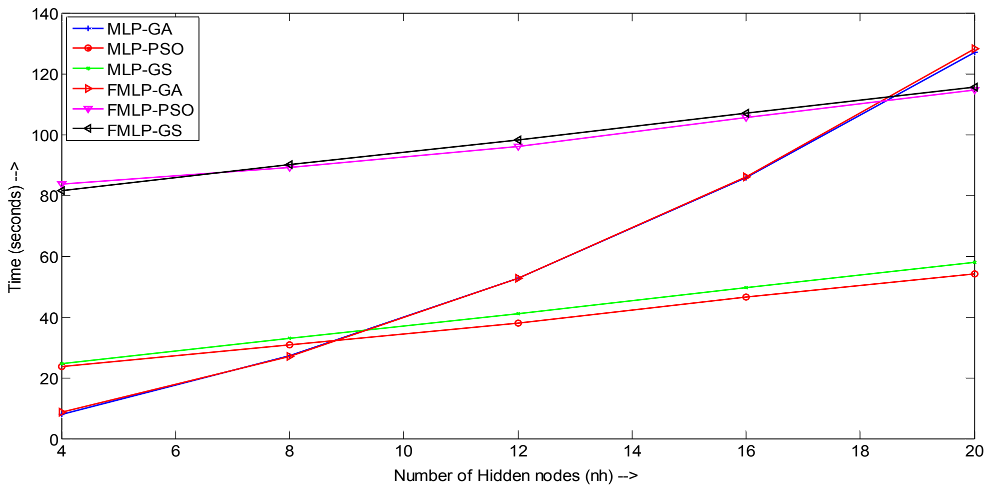


Figure 3.12 Effect of number of hidden nodes on average testing time

Table 3.6(a). t-test for GA based Fuzzy MLP

Dataset	t-value	Mean Difference	95% Confidence Interval of the Difference	
			Lower	Upper
XOR4	4.242641	12	4.1470	19.852
Iris	606.5	97.04	96.595	97.484
BNA	5.45E+16	99.9	99.9	99.9
BT	88.68185	94.96	91.987	97.933
HR	5.09E+16	93.4	93.4	93.4
TAE	119.1275	51.1	49.909	52.290
UKM	116.2636	88.506	86.392	90.619

Table 3.6(b). t-test for PSO based Fuzzy MLP

Dataset	t-value	Mean Difference	95% Confidence Interval of the Difference	
			Lower	Upper
XOR4	4.242641	12	4.1470	19.852
Iris	61.16098	95.194	90.872	99.515
BNA	25.41472	85.06	75.767	94.352
BT	24.47444	61.52	54.541	68.498
HR	24.54471	61.516	54.557	68.474
TAE	137.7979	63.18	61.907	64.452
UKM	54.96415	93.06	88.359	97.760

Table 3.6(c). t-test for GS based Fuzzy MLP

Dataset	t-value	Mean Difference	95% Confidence Interval of the Difference	
			Lower	Upper
XOR4	6.531973	80	45.995	114.00
Iris	5.355325	56.266	27.095	85.436
BNA	21.17808	58.44	50.778	66.101
BT	14.95018	68.12	55.469	80.770
HR	5.308021	46.296	22.080	70.511
TAE	21.06654	59.06	51.276	66.843
UKM	7.834265	73.4	47.387	99.412

3.5. Conclusion

In this work, fuzzy MLP has been successfully applied for pattern classification, which is trained with three different population based optimization algorithms, GA, PSO and GS. The trained network is tested with one bit parity dataset and six UCI datasets. The results of both MLP and Fuzzy MLP are compared with each other based on hidden layer setting, classification accuracy and classification time. Statistical analysis (ANOVA) of all the results is carried out using SPSS software which mentions t-value of results against each dataset. The experimental result showed that GA based fuzzy MLP could classify data with an average accuracy of 82.57% for the considered datasets. Future research includes simultaneous improvement of the proposed Fuzzy MLP using new optimization algorithms and testing the performance of the model with larger and number of datasets.

References

[1] D.J. Montana, L. Davis, "Training Feed forward Neural Networks Using Genetic Algorithms," Machine Learning, pp. 762-767.

[2] S. Mitra, R.K. De, S.K. Pal, "Knowledge-Based Fuzzy MLP for Classification and Rule Generation," IEEE Trans. on Neural Networks, Vol. 8, No. 6, November 1997, pp. 1338-1350.

[3] L. Liu, X. Peng, “An Improved Diagonal Recurrent Neural Network and Its Application in Nonlinear Time Series Forecasting,” Information Engineering Letters, volume 2, number 1, March, 2012, pp. 18-25.
[4] T. Quah, B. Srinivasan, “Improving returns on stock investment through neural network selection,” Expert Systems with Applications 17 (1999) 295–301.
[5] M. Karabataka, M.C. Ince, “An expert system for detection of breast cancer based on association rules and neural network,” Expert Systems with Applications 36 (2009) 3465–3469.
[6] W. Sibanda, P. Pretorious, “Novel Application of Multi-Layer Perceptrons (MLP) Neural Networks to Model HIV in South Africa using Seroprevalence Data from Antenatal Clinics,” International Journal of Computer Applications, 35(5), December 2011, pp. 26-31.
[7] P.K. Dash, H.S. Behera, I.W.C. Lee, “Time sequence data mining using time-frequency analysis and soft computing techniques,” Applied Soft Computing 8(1):202:215 (2008).
[8] M. Uyara, S. Yildirim, M.T. Gencoglu, “An expert system based on S-transform and neural network for automatic classification of power quality disturbances,” Expert Systems with Applications 36 (2009); 5962–5975.
[9] H.S. Behera, P.K. Dash, B.N. Biswal, “Power quality time series data mining using S-transform and fuzzy expert system,” Applied Soft Computing 10(3): 945-955 (2010).
[10] C.L. Liu, F. Yin, D.H. Wang, Q.F. Wang, “Online and offline handwritten Chinese character recognition: benchmarking on new databases,” Pattern Recognition 46(1): 155-162 (2013).
[11] T.H. Hildebrandt, W. Liu, “Optical recognition of Chinese characters: advances since 1980,” Pattern Recognition 26(2): 205-225 (1993).
[12] C.L. Liu, S. Jaeger, M. Nakagawa, “Online recognition of Chinese characters: the state-of-the-art,” IEEE Trans. on Pattern Analysis and Machine Intelligence, 26(2), pp. 198-213 (2004).
[13] J.F. Vargas, M.A. Ferrer, C.M. Travieso, J.B. Alonso, “Off-line signature verification based on grey level information using texture features”, Pattern Recognition, 44(2), 375-385.
[14] T. Dash, T. Nayak, S. Chattopadhyay, “Offline Verification of Hand Written Signature using Adaptive Resonance Theory Net (Type-1),” International Journal of Signal Processing Systems, 1(1), pp. 17-22.
[15] J.C. Yang, D.S. Park, “A fingerprint verification algorithm based on tessellated invariant moment features,” Neurocomputing, 71(2008), 1939-1946.
[16] P. Smaragdis, B. Raj, “The Markov selection model for concurrent speech recognition,” Neurocomputing, 80, 64-72.
[17] P.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, 1973, New York.
[18] R.P. Lippmann, “Pattern classification using neural networks,” IEEE Commun. Mag., pp. 47–64, Nov. 1989.

[19] G.P. Zhang, "Neural Networks for Classification: A Survey," IEEE Trans. on Systems, Man, and Cybernetics—Part C: Applications and Reviews, Vol. 30, No. 4, November 2000**,** pp. 451-462.

[20] K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks, vol. 4, pp. 251–257, 1991.

[21] G. Cybenko, "Approximation by superpositions of a sigmoidal function," Math. Contr. Signals Syst., vol. 2, pp. 303–314, 1989.

[22] M. Kang, D.P. Brown, "A modern learning adaptive functional neural network applied to handwritten digit recognition," Information Sciences 178(2008), 3802–3812.

[23] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, Michigan (1975); re-issued by MIT Press (1992).

[24] R. Eberhart, J. Kennedym, "A new optimization using particle swarm theory," Micro machine and Human Science, 1995. MHS'95, In: Sixth International Symposium on, pp. 39-43. IEEE, 1995.

[25] R. Eberhart, Y. Shi, "Particle Swarm Optimization: developments, applications and resources," In: congress on evolutionary computation 2001 IEEE service center.

[26] E. Rashedi, Gravitational Search Algorithm [M.Sc. Thesis], Shahid Banohar University of Kerman, Kerman, Iran, 2007.

[27] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, "GSA: A Gravitational Search Algorithm," Information Sciences 179(2009), 2232-2248.

[28] P.G. Espejo, S. Ventura, F. Herrera, "A Survey on the Application of Genetic Programming to Classification," IEEE Trans. on Systems, Man, and Cybernetics—Part C: Applications And Reviews, Vol. 40, No. 2, March 2010, pp. 121-144.

[29] A.R. Jordehi, J. Jasni, "Parameter selection in particle swarm optimization: a survey," J Experimental & Theoretical Artificial Intelligence, 25:4, 527-542.

[30] S. Dehuria, R. Roy, S.B. Cho, A. Ghosh, "An improved swarm optimized functional link artificial neural network (ISO-FLANN) for classification," The Journal of Systems and Software 85 (2012) 1333–1345.

[31] S.J. Coakes, S. Lyndall, SPSS: Analysis without anguish using SPSS version 14.0 for Windows, John Wiley & Sons, Inc., 2009.

Appendix-3.1 Comparison of MLP and Fuzzy MLP with various hidden layer settings

Dataset →				XOR4					Iris					BNA					BT				
# of Hidden nodes (*nh*) →				4	8	12	16	20	4	8	12	16	20	4	8	12	16	20	4	8	12	16	20
Mean Squared Error (MSE)	MLP	GA	±std	0.002	0.001	0.0009	0.0006	0.0003	0.003	0.002	0.001	0.001	0.004	9.030	0.004	0.004	0.003	0.003	0.001	0.001	0.002	0.003	0.003
			Mean	0.251	0.248	0.248	0.248	0.248	0.061	0.056	0.055	0.055	0.054	1.340	0.016	0.016	0.016	0.016	0.038	0.037	0.037	0.037	0.037
			Min	0.249	0.246	0.246	0.247	0.248	0.059	0.055	0.054	0.054	0.054	0.016	0.016	0.016	0.016	0.016	0.037	0.037	0.037	0.037	0.037
		PSO	±std	0.080	0.122	0.133	0.162	0.150	0.142	0.176	0.194	0.220	0.211	0.149	0.180	0.199	0.206	0.222	0.069	0.080	0.121	0.132	0.139
			Mean	0.466	0.418	0.476	0.416	0.480	0.218	0.139	0.171	0.195	0.177	0.143	0.124	0.181	0.140	0.102	0.319	0.319	0.327	0.325	0.327
			Min	0.441	0.362	0.442	0.332	0.444	0.168	0.087	0.108	0.113	0.102	0.083	0.062	0.120	0.084	0.036	0.305	0.303	0.304	0.300	0.303
		GS	±std	0.069	0.112	0.118	0.138	0.150	0.086	0.119	0.157	0.152	0.137	0.047	0.071	0.085	0.071	0.098	0.058	0.076	0.106	0.127	0.132
			Mean	0.514	0.515	0.538	0.547	0.504	0.449	0.410	0.350	0.471	0.562	0.309	0.317	0.328	0.474	0.244	0.362	0.376	0.371	0.372	0.393
			Min	0.505	0.494	0.520	0.523	0.475	0.427	0.386	0.312	0.429	0.549	0.304	0.309	0.315	0.467	0.232	0.355	0.368	0.359	0.358	0.377
	Fuzzy MLP	GA	±std	0.003	0.005	0.008	0.009	0.008	0.003	0.001	0.002	0.001	0.001	0.003	0.002	0.001	0.001	0.001	0.002	0.002	0.001	0.001	0.001
			Mean	0.127	0.129	0.136	0.142	0.150	0.018	0.018	0.018	0.019	0.020	0.009	0.009	0.010	0.011	0.012	0.005	0.003	0.003	0.002	0.002
			Min	0.125	0.125	0.126	0.130	0.137	0.000	0.018	0.018	0.018	0.018	0.009	0.009	0.009	0.010	0.010	0.004	0.002	0.002	0.002	0.002
		PSO	±std	0.171	0.194	0.206	0.222	0.220	0.111	0.116	0.132	0.146	0.145	0.090	0.123	0.125	0.128	0.137	0.090	0.115	0.127	0.141	0.146
			Mean	0.057	0.044	0.037	0.049	0.042	0.059	0.067	0.042	0.051	0.082	0.121	0.098	0.090	0.124	0.104	0.087	0.108	0.105	0.058	0.067
			Min	0.003	0.003	0.004	0.004	0.003	0.019	0.034	0.010	0.010	0.030	0.080	0.043	0.046	0.074	0.043	0.049	0.060	0.061	0.013	0.021
		GS	±std	0.099	0.162	0.178	0.125	0.151	0.061	0.085	0.088	0.074	0.088	0.047	0.071	0.085	0.071	0.098	0.042	0.069	0.089	0.097	0.103
			Mean	0.297	0.161	0.227	0.719	0.512	0.276	0.259	0.340	0.502	0.448	0.309	0.317	0.328	0.474	0.244	0.304	0.298	0.302	0.312	0.341
			Min	0.279	0.131	0.191	0.707	0.495	0.255	0.240	0.327	0.493	0.437	0.304	0.309	0.315	0.467	0.232	0.298	0.292	0.284	0.298	0.320

Appendix-3.1 Comparison of MLP and Fuzzy MLP with various hidden layer settings (contd..)

Dataset →	# of Hidden nodes (*nh*) →	Mean Squared Error (MSE)																	
		Fuzzy MLP									MLP								
		GS			PSO			GA			GS			PSO			GA		
		Min	Mean	±std	Min	Mean	±std	Min	Mean	±std	Min	Mean	±std	Min	Mean	±std	Min	Mean	±std
HR	4	0.497	0.505	0.061	0.434	0.452	0.066	0.013	0.014	0.002	0.631	0.641	0.052	0.490	0.514	0.075	0.049	0.087	0.09
	8	0.501	0.512	0.095	0.431	0.456	0.106	0.011	0.012	0.001	0.573	0.591	0.093	0.506	0.530	0.104	0.06	0.108	0.115
	12	0.853	0.861	0.079	0.427	0.454	0.120	0.011	0.012	0.001	0.622	0.641	0.111	0.494	0.529	0.131	0.061	0.105	0.127
	16	0.713	0.725	0.098	0.436	0.464	0.126	0.011	0.011	0.003	0.668	0.688	0.124	0.502	0.532	0.137	0.013	0.058	0.141
	20	0.801	0.810	0.089	0.414	0.453	0.133	0.011	0.011	0.003	0.685	0.706	0.125	0.490	0.527	0.149	0.021	0.067	0.146
TAE	4	0.247	0.250	0.034	0.434	0.452	0.066	0.019	0.019	0.001	0.653	0.661	0.062	0.490	0.514	0.075	0.080	0.082	0.003
	8	0.248	0.254	0.052	0.431	0.456	0.106	0.019	0.019	0.004	0.654	0.669	0.100	0.506	0.530	0.104	0.077	0.078	0.001
	12	0.410	0.414	0.043	0.427	0.454	0.120	0.019	0.019	0.002	0.592	0.612	0.129	0.494	0.529	0.131	0.077	0.078	0.001
	16	0.411	0.415	0.044	0.436	0.464	0.126	0.019	0.019	0.003	0.972	0.984	0.099	0.502	0.532	0.137	0.077	0.077	0.003
	20	0.296	0.310	0.059	0.414	0.453	0.133	0.019	0.019	0.004	0.934	0.946	0.105	0.490	0.527	0.149	0.077	0.077	0.002
UKM	4	0.241	0.245	0.033	0.186	0.199	0.040	0.034	0.036	0.003	0.124	0.128	0.027	0.088	0.095	0.033	0.035	0.039	0.006
	8	0.238	0.243	0.047	0.184	0.194	0.047	0.033	0.039	0.006	0.116	0.121	0.035	0.088	0.096	0.042	0.036	0.047	0.008
	12	0.249	0.256	0.054	0.182	0.196	0.062	0.037	0.046	0.006	0.105	0.114	0.049	0.089	0.100	0.054	0.043	0.054	0.006
	16	0.275	0.280	0.054	0.181	0.195	0.065	0.043	0.051	0.005	0.110	0.119	0.054	0.085	0.096	0.056	0.055	0.060	0.002
	20	0.386	0.391	0.044	0.182	0.199	0.067	0.051	0.056	0.003	0.248	0.252	0.040	0.091	0.101	0.058	0.060	0.062	0.001

Chapter-4

4

Hybrid Gravitational Search and Particle Swarm based Fuzzy MLP for Medical Data Classification

Preview:

In this chapter, a hybrid of GS and PSO, called GSPSO, has been proposed which improves the performance of the algorithms given in chapter 3. Proposed GSPSO algorithm has been tested for medical datasets and corresponding performances of the algorithms are compared for each datasets.

4.1. Motivation

In recent years, the incorporation of soft computing approaches in medical diagnosis has achieved a new tendency to be employed successfully in a large number of medical applications. Many of the medical diagnosis procedures can be grouped into intelligent classification tasks [1]. These classification procedures can be (i) binary classification, where data is separated between only two classes, (ii) multi-class classification, where data is separated among more than two classes. For example, classifying a diabetic patient is a binary classification task, where the patient may be suffering from diabetes mellitus or diabetes insipidus. Similarly, detection of lung cancer is a type of multi-class classification based problem However, classification accuracy in medical datasets are still less to be adopted in medical practice. Hence, in this work, an attempt has been made to improve the performance of such classifier model by developing new optimization algorithms.

4.2. Related works

In computer aided medical research, many researchers have tried to use different methods to improve data classification accuracy. Methods with better classification accuracy will provide more sufficient information to identify the potential patients and to improve the diagnosis accuracy [1]. Medical database classification is not just a kind of classification problem rather is a kind of complex optimization problem whose goal is not only to find an optimal solution but also to provide accurate diagnosis for diseases. And therefore, meta-heuristic algorithms such as genetic algorithm, particle swarm optimization etc. and soft computing and machine learning tools such as neural networks, decision tree, and fuzzy set theory have been successfully applied in this area and have achieved significant results [2]. Artificial neural network (ANN) based approach aims to provide a filter that distinguishes the cases which do not have disease, therefore reducing the cost of medication and overheads of doctors. Back propagation neural network (BPNN) was used by Floyd et al. [3] for classification of medical data and this work achieved an overall accuracy of 50%. Wu et al. [4] used similar BPNN technique for application in breast tumor identification. In this work, they used 10 hidden nodes in the ANN and tested their algorithm in a small database 133 instances. The performance of their approach outperformed domain expert decisions. In another study, rule extraction from ANN has been employed for prediction of breast cancer from Wisconsin dataset [5,6]. All the above methods used back propagation learning for training the ANN, where solution got trapped in the local minima. Therefore, Fogel et al. [7] attempted to solve the medical database classification problem using evolutionary computation and could achieve higher prediction accuracy than the above techniques. However, this work suffered from higher computational cost in application. Therefore, researchers tried to use an integrated fuzzy rule based approach to solve the above problem [1,8].

Many researchers have used fuzzy classifier tool [8,9] which employ fuzzy rule base with artificial intelligence models [10,11] to extract fuzzy rules directly from database. Gadaras and Mikhailov [12] presented a novel fuzzy classification framework for medical data classification. Their approach extracted rules from labeled numerical data and the results showed excellent outcomes after testing the method in three medical datasets. Fernandez et al. [13] analyzed the behavior of fuzzy rule based classification systems in framework of imbalanced datasets. A hybrid classification fuzzy model was proposed by [8] which used two algorithms, modified Gath-Geva and C4.5. The former algorithm was used for function estimation and the later one was used for classification problems. The above techniques could fail for unlabelled or mislabeled databases, therefore researches tried to use fuzzy rule based techniques with other soft computing models such as neural networks [10,11].

Data mining techniques such as ontology based intelligent systems [14], discriminant analysis [10], and least square SVM [15] have been applied in medical database classification. However, accuracies in current methods are still low and insignificant enough to be adopted in medical practice. In this research, our contribution is to develop a hybrid model combining evolutionary computation, fuzzy logic and neural network to maximize the classification accuracy and decision taking speed. In the proposed hybrid model, we combined Gravitational Search (GS) technique [16] with Particle Swarm Optimization (PSO) [17] to train Fuzzy Multilayer perceptron (Fuzzy MLP) for medical data classification. The proposed algorithm has been tested for classification of five medical datasets and the results so obtained have been compared with GS based Fuzzy MLP and PSO based Fuzzy MLP.

4.3. Proposed Methodology

This section describes the Fuzzy MLP architecture along with the proposed hybrid training algorithm for medical data classification.

4.3.1. Fuzzy MLP Architecture

The proposed Fuzzy Multilayer Perceptron (Fuzzy MLP) [18] is n–layer architecture, where $n>3$. The layers are (i) input layer(IL), (ii) output layer(OL) and (iii) hidden layer(HL). It is a feedforward neural network, where non-linear elements, called neurons are arranged in layers. The data flow from the input layer to the output layer via hidden layer(s) and intermediate processing. Learning is supervised for Fuzzy MLP where the computed error at the output layer acts as the supervisor for Fuzzy MLP learning. The proposed neural network is trained with three different evolutionary algorithms, (i) Gravitational Search (GS), (ii) Particle Swarm Optimization (PSO), and (iii) Hybrid of GS and PSO called *GSPSO*. The pseudo-code and governing equations for the proposed GSPSO algorithm has been given in next subsection.

Data fuzzification: The first phase of the method is fuzzification of the input data which can be done by use of a fuzzy membership function (MF). In fuzzy set theory, there exists no perfect rule for selecting a fuzzy MF. Researches always consider different MF for different problems. This work will adopt spline based or *S*-shaped function as primary MF. This MF puts the input dataset in a range *[a,b]*.The equation for *S*-shaped MF has been given in Equation-1.

$$f(x,a,b) = \begin{Bmatrix} 0, x < a \\ 2\left(\frac{x-a}{b-a}\right)^2, a \leq x \leq \frac{a+b}{2} \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2, \frac{a+b}{2} \leq x \leq b \\ 1, x > b \end{Bmatrix} \tag{1}$$

4.3.2. Proposed GSPSO Algorithm

Gravitational search (GS) technique [16] is best suited when the search space is large due to its local search capability [19]. In other hand, PSO is well known for its social movements. In GSPSO algorithm, following major steps are followed.

1. All agents (particle position, agent mass, acceleration) are initialized.
2. Calculate gravitational force acting on each particle due to mutual attraction.
3. Calculate acceleration of each particle due to force of attraction.
4. Update the best solution with new position of the agents (candidate solutions).
5. If stopping criterion is satisfied, best solution is already obtained; else repeat the steps 2-5.

The following equations are used for various calculations during the above steps. The gravitational constant (G) is calculated using Equation-2. In the following equation, α is a small constant, G_0 is the initial gravitational constant, *iterator* is the current iteration, *maxiteration* is maximum iteration to be performed.

$$G(t) = G_0 \times \exp\left(-\propto \times \frac{iterator}{\text{maxiteration}}\right) \tag{2}$$

Fitness of the candidate solution in this work is calculated by considering the mean squared error (MSE) as given in Equation-3, where *N* is the total number of training instances in the medical dataset, *Target* is the target class for the current instance and *Output* is the computed output class for the current instance.

$$Fitness = \text{MSE} = \sum_{\text{i}=1}^{\text{N}} (\text{Target(i)} - \text{Output(i)})^2 \tag{3}$$

Mass of a particle is calculated by using Equation-4, where fit_i is the fitness value of agent i, in the current epoch evaluation and *worst* is defined as given in Equation-4 below.

$$M_i = \frac{fit_i - worst}{\sum_{j=1}^{n}(fit_i - worst)} \tag{4}$$

The force acting on object *i* from object *j* is defined as given in Equation-5. Similarly, summing up all the individual forces acting on particle *i,* we will get the resultant force on the particle *i* as given in Equation-6. In the following equations, R_{ij} is defined as the Euclidean distance between two objects *i* and *j*. The parameters r_1, r_2 and θ are arbitrary random values between 0 and 1.

$$F_{ij} = G_0\left(\frac{M_i \times M_j}{R_{ij} + \theta}\right)(x_j - x_i) \tag{5}$$

$$F_i = \sum_{j=1, j \neq i}^{n} r_j F_{ij} \tag{6}$$

Therefore, resultant force acting on particle *i* can be written as follows,

$$F_i = G_0 M_i \sum_{j=1, j \neq i}^{n} \frac{M_j}{R_{ij}+\theta} (x_j - x_i) \tag{7}$$

Acceleration of object *i* is calculated using Equation-8.

$$a_i = \frac{F_i}{M_i} \tag{8}$$

Velocity and position of particle *i* are calculated using Equations-9 and Equation-10 respectively, where *$V_i(t)$* is the velocity of agent *i* at iteration *t*, *c'_1* and *c'_2* are acceleration coefficients, *gbest* is the best fitness so far, *$X_i(t)$* is the position of particle in iteration $_t$, a_i is the acceleration of particle *i*, θ_1 and θ_2 are random numbers between 0 and 1.

$$V_i(t+1) = w \times V_i(t) + c_1' \times \theta_1 \times a_i c_i(t) + c_2' \times \theta_2 \times (gbest - X_i(t)) \tag{9}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{10}$$

4.3.3. GSPSO Pseudo-code for training Fuzzy MLP

```
Input medical dataset
Fuzzify the input dataset using member function
Initialize Fuzzy MLP parameters based on input data: number of input nodes,
number of output nodes, number of hidden nodes, weights, biases
Initialize GSPSO algorithm parameters: maximum iteration, gravitational
consant, particle initial position, inertia weights, $c_1$, $c_2$, search space dimension
Set iterator=0
While iterator < Maximum iteration
   iterator=iterator+1
   Update G using Equation-2
   Set gravitational force (F), mass of each particle (M), position of particle (X),
   acceleration (a) to 0
   For each agent
        Initialize weights (W) and biases (b) of Fuzzy MLP
   End for
   Calculate fitness using Equation-3
   If(obtained fitness is better than gbest)
        set gbest to obtained fitness
   End if
   Update mass of particle using Equation-4
   Calculate gravitational forces acting on each particle and corresponding
   resultant force using Equations-5,6,7
   Calculate acceleration of particle using Equation-8
   Update velocity of particle using Equation-9
   Update new position of particle using Equation-10
End while
```

4.4. Experimental Results

Following subsections describes the experimental details and performance analysis of the proposed model.

4.4.1. Dataset Description

The proposed model has been tested with five medical datasets obtained from UCI machine learning repository [20]. The following table shows a summary of five medical datasets which are used for testing purpose in this work. Number of attributes in Table-4.1 is the number of input attributes plus the class attribute. These datasets are (i) Wisconsin breast cancer (WBC), (ii) Heart disease, (iii) Hepatitis, (iv) Indian liver patient database (ILPD), and (v) lung cancer dataset. Further details and properties of each dataset can be obtained from [19].

Table 4.1. UCI Medical dataset properties

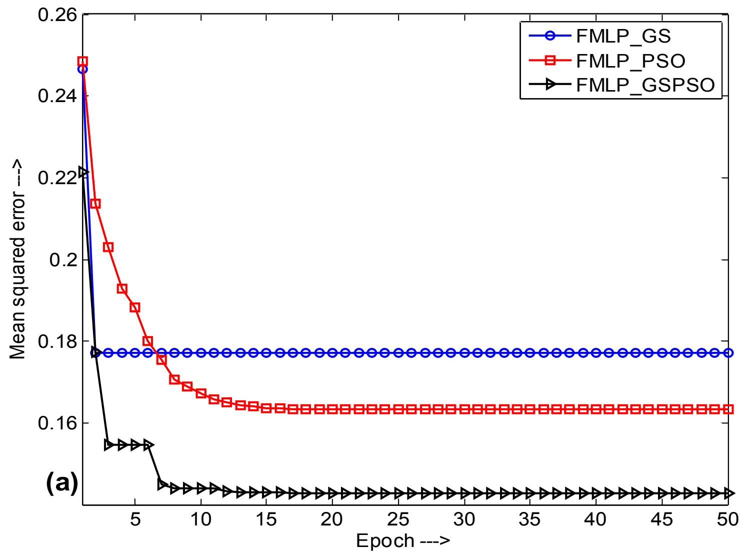
Dataset	No. of instances	No. of attributes	No. of classes
WBC	699	11	2
Heart disease	270	14	2
Hepatitis	155	20	2
ILPD	583	10	2
Lung cancer	32	57	3

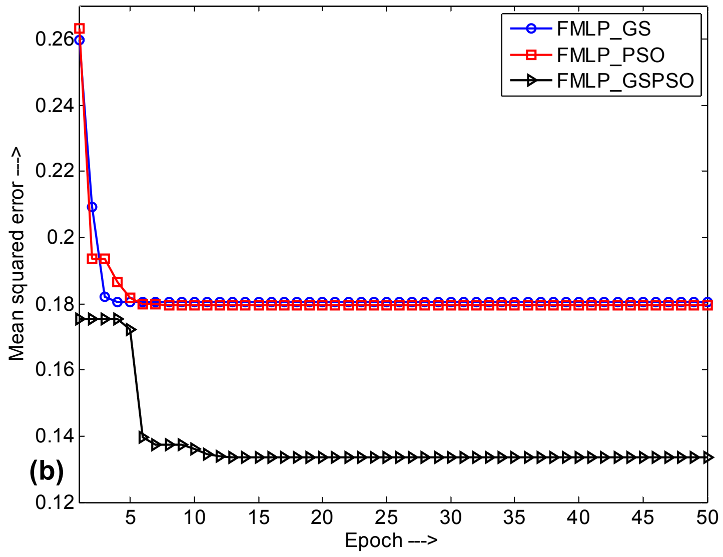
4.4.2. Experimental setup and Simulation Parameters

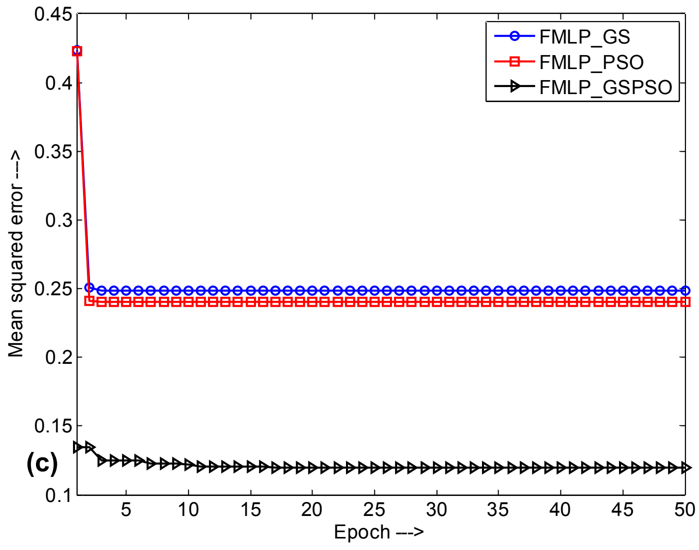
All the simulations are carried out in MATLAB R2010a which is installed in a PC having Windows 7 OS and 2 GB main memory. The processor is Intel dual core and each processor has an equal computation speed of 2 GHz (approx.)

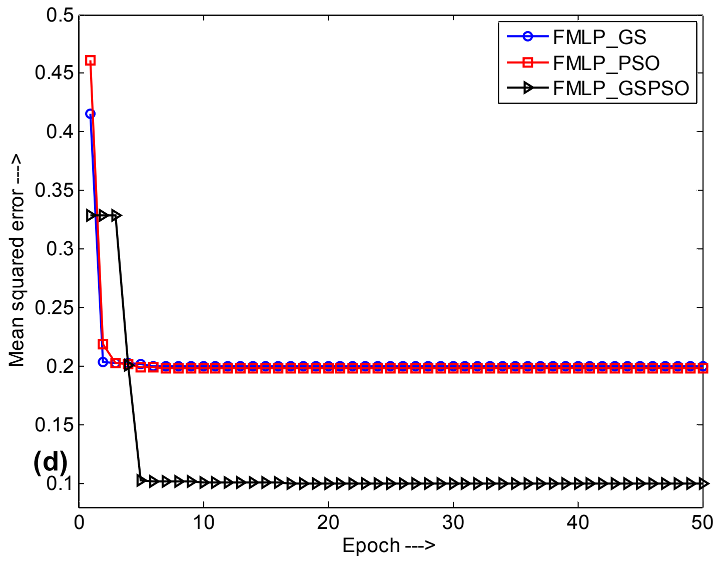
Number of hidden units in the Fuzzy MLP is set to *2n,* where n is the total number of input units. Number of population is 30 for all the cases. Inertia weight (w) is set to 2; w_{max} and w_{min} are set to 0.9 and 0.5 respectively. The coefficients, c_1 and c_2 are set to 2 each; gravitational constant (G_0) is set to 1. All the simulations are allowed to run for 50 epochs. For all the three models, GS based Fuzzy MLP (Fuzzy MLP-GS), PSO based Fuzzy MLP (Fuzzy MLP-PSO) and proposed GSPSO based Fuzzy MLP (Fuzzy MLP-GSPSO), MSE is noted against each epoch during training. Testing results are classification accuracy, simulation time. Following tables shows the simulation results for all the five tested datasets. All the three models are compared for the above four parameters.

Plots given in Fig. 4.1(a)-4.1(e) is a comparison of all the three models based on convergence of error during training. Table-4.2 shows mean MSE obtained by all the three models for various datasets. It can be seen that the proposed GSPSO based Fuzzy MLP outperforms the other two models for all the datasets except lung cancer dataset. However, the classification accuracy obtained by the proposed model for the lung cancer dataset is higher than that obtained by the GS and PSO models (see Table-4.3).


FMLP_GS
FMLP_PSO
FMLP_GSPSO
Mean squared error --->
Epoch --->
(a)


FMLP_GS
FMLP_PSO
FMLP_GSPSO
Mean squared error --->
Epoch --->
(b)


FMLP_GS
FMLP_PSO
FMLP_GSPSO
Mean squared error --->
Epoch --->
(c)



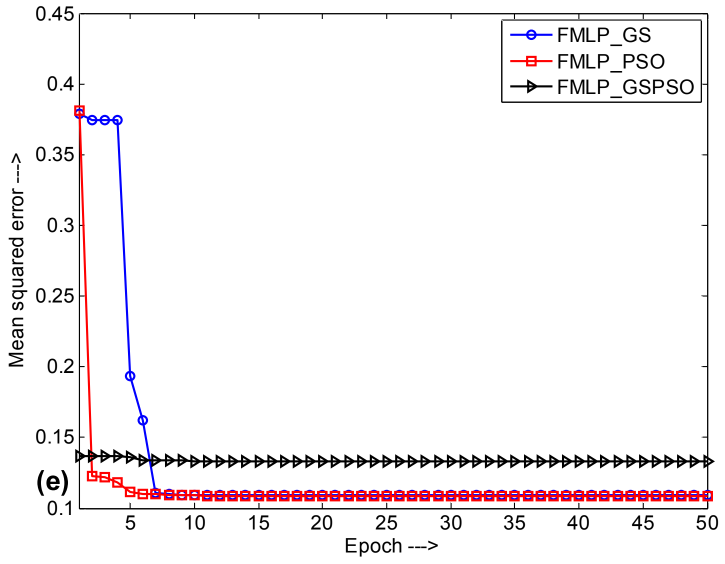


Figure 4.1. Convergence curves of Fuzzy MLP-GS, Fuzzy MLP-PSO, and Fuzzy MLP-GSPSO for five medical datasets: (a)WBC (b) Heart disease (c) Hepatitis (d) ILPD (e) Lung cancer dataset

The simulation time is also a crucial parameter for comparison which is presented in Table-4.4. Proposed GSPSO model achieved the best results i.e. 82% accuracy for the WBC dataset and 81% accuracy for ILPD datasets. However, it could achieve only 67% accuracy when tested for hepatitis dataset. It could be concluded that the proposed GSPSO model outperforms GS and PSO model for these five medical datasets. And hence could be adopted in medical diagnosis in practice.

Fuzzy MLP-PSO has better training speed than the Fuzzy MLP-GS and Fuzzy MLP-GSPSO. However, as training is carried out before adopting the model physically, therefore, it is of least importance. The real decision speed is dependent on the testing time, which is the time taken by the models to classify the input instance. Table-4.5 also shows that the proposed

GSPSO model could achieve the best result within 0.5 seconds of time, which is also an advantage of adopting the proposed model in medical practices.

Table 4.2. MSE (mean±std. deviation) for five tested medical datasets

Dataset	Fuzzy MLP-GS	Fuzzy MLP-PSO	Fuzzy MLP-GSPSO
WBC	0.1785±0.0098	0.1690±0.0155	0.1461±0.0123
Heart disease	0.1826±0.0118	0.1819±0.0121	0.1331±0.0124
Hepatitis	0.2520±0.0248	0.2442±0.0258	0.2212±0.0032
ILPD	0.2047±0.0305	0.2043±0.0372	0.1164±0.0560
Lung cancer	0.1332±0.0735	0.1150±0.0386	0.1333±0.0012

Table 4.3. Classification accuracy (mean±std. deviation) for five tested medical datasets

Dataset	Fuzzy MLP-GS	Fuzzy MLP-PSO	Fuzzy MLP-GSPSO
WBC	72.6960±5.621	75.5700±4.597	**81.6953±2.001**
Heart disease	76.2987±0.004	79.2963±0.020	76.9697±1.350
Hepatitis	46.2885±4.665	52.4186±0.320	66.3700±2.740
ILPD	67.6150±3.721	67.5800±0.001	80.6600±6.330
Lung cancer	47.6525±8.976	50.0000±0.000	71.8750±0.000

Table 4.4. Simulation time (training time+testing time) for five tested medical datasets

Dataset	Fuzzy MLP-GS	Fuzzy MLP-PSO	Fuzzy MLP-GSPSO
WBC	115.94+0.027	38.990+0.026	114.48+0.026
Heart disease	56.530+0.015	17.266+0.013	57.750+0.011
Hepatitis	44.700+0.008	12.880+0.008	45.300+0.008
ILPD	100.60+0.022	33.610+0.022	102.50+0.024
Lung cancer	43.790+0.007	10.720+0.006	43.750+0.006

4.5. Conclusion and Future Works

In this work, a new hybrid of gravitational search and particle swarm called GSPSO has been proposed for classification medical data using Fuzzy MLP. Five benchmark medical datasets: breast cancer, heart disease, hepatitis, liver diseases, and lung cancer, are used to evaluate the performance of the proposed model. The results are compared with GS and PSO based Fuzzy MLP models. For all the datasets, the GSPSO model shows better performance in terms of error convergence, classification accuracy and decision speed.

It will be interesting to employ the proposed model in robotics and manufacturing fields which include data classifications. The authors are currently working on this area.

References

[1] C.-Y. Fan, P.-C. Chang, J.-J. Lin, J.C. Hsieh, "A hybrid model combining case-based reasoning and fuzzy decision tree for medical data classification," Applied Soft Computing 11, 632-644 (2011).

[2] C.-C. Bojarczuk, H.-S. Lopes, A,-A. Freitas, "Genetic programming for knowledge discovery in chest-pain diagnosis," IEEE Engineering in Medicine and Biology Magazine 19(4) 38-44 (2000).

[3] C.E. Floyd, J.Y. Lo, A.J. Yun, D.C. Sullivan, P.J. Kornguth, "Prediction of breast cancer malignancy using an artificial neural network," Cancer 74, pp. 2944-2998 (1994).

[4] Y.-Z. Wu, M.-L. Giger, K. Doi, C.J. Vyborny, R.A. Schmidt, C.E. Metz, "Artificial neural networks in mammography: application and decision making in the diagnosis of breast cancer," Radiology 187, pp. 81-87 (1993).

[5] R. Setiono, L. Huan, "Understanding neural networks via rule extraction," In: proceedings of the International Joint Conference on Artificial Intelligence, Morgan Kauffman, San Mateo, CA, pp. 480-487 (1995).

[6] R. Setiono, "Generating concise and accurate classification rules for breast cancer diagnosis," Artificial Intelligence in Medicine 18, 205-219 (2000).

[7] D.B. Fogel, E.C. Wasson, E.M. Boughton, "Evolving neural networks for detecting breast cancer," Cancer Letters 96(1) 49-53 (1995).

[8] P. Pulkkinen, H. Koivisto, "Identification of interpretable and accurate fuzzy classifiers and function estimators with hybrid methods," Applied Soft Computing 7, 520-533 (2007).

[9] P.C. Chang, T.W. Liao, "Combining SOM and fuzzy rule base for flow time prediction in semiconductor manufacturing factory," Applied Soft Computing 6(2), 198-206 (2006).

[10] X.-N. Song, Y.-J. Zheng, X.-J. Wud, X.-B. Yang, J.-Y. Yang, "A complete fuzzy discriminant analysis approach for face recognition," Applied Soft Computing 10, 208-214 (2010).

[11] L.B. Goncalves, M.M.B.R. Vellasco, M.A.C. Pacheco, F.J.D. Souja, "Inverted hierarchical neuro-fuzzy BSP system: a novel neuro-fuzzy model for pattern classification and rule extraction in databases," IEEE Transaction on Systems, Man, and Cybernetics Part C: Applications and reviews 36(2), 236-248 (2006).

[12] I. Gadaras, L. Mikhailov, "An interpretable fuzzy rule-based classification methodology for medical diagnosis," Artificial Intelligence in Medicine 47(1), 25-41 (2009).

[13] A. Fernandez, M.J. Jesus, F. Herrera, "On the influence of an adaptive inference system in fuzzy rule based classification systems for imbalanced datasets," Expert Systems with Applications 36, 9805-9812 (2009).

[14] C.S. Lee, M.H. Wang, "Ontology-based intelligent healthcare agent and its application to respiratory waveform recognition," Expert Systems with Applications 33(3), 606-619 (2007).

[15] K. Polat, S. Gunes, A. Arslan, "A cascade learning system for classification of diabetes disease: generalized discriminant analysis and least square support vector machine," Expert Systems with Applications 34(1), 482-487 (2008).

[16] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, "GSA: A Gravitational Search Algorithm," Information Sciences 179, 2232-2248 (2009).

[17] R. Eberhart, J. Kennedym, “A new optimization using particle swarm theory,” Micro machine and Human Science, MHS’95, In: Sixth International Symposium on, pp. 39-43. IEEE (1995).

[18] S. Mitra, R. K. De, S. K. Pal, “Knowledge-Based Fuzzy MLP for Classification and Rule Generation,” in IEEE Transactions on Neural Networks, Vol. 8, No. 6, November 1997, pp. 1338-1350.

[19] S. Mirjalili, S.Z.M. Hashim, H.M. Sardroudi, “Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm,” Applied Mathematics and Computing 218, 11125-11137 (2012).

[20] K. Bache, M. Lichman, “UCI machine learning repository [http://archive.ics.uci.edu/ml] Irvine, CA,” University of California, School of Information and Computer Science, 2013.

Chapter-5

5

Conclusion and Future Directions

Preview:

In this chapter, a summary of finding of the thesis works has been presented. Further directions to continue the research has been given.

Conclusion:

In this thesis, the Fuzzy MLP for pattern classification has been applied for nonlinear pattern classification problems and over a large number of datasets. Various optimization algorithms have been employed for training the Fuzzy MLP neural network. Performance analysis, both training and testing, has been extensively carried out to observe the efficiency and effectiveness of the proposed models. A set of conclusions obtained from the contributions in previous chapters are outlined below.

- Fuzzy MLP outperforms MLP for all the tested pattern classification problems viz. UCI datasets
- GA based fuzzy MLP could classify data with an average accuracy of 82.57% for the input datasets.
- Proposed GSPSO based Fuzzy MLP has been applied to problems of medical data classification and tested with five benchmark datasets obtained from UCI machine learning repository. The GSPSO model shows better performance in terms of error convergence, classification accuracy and decision speed for all these datasets and therefore, the proposed model could be adopted in medical practice.

Future directions:

As future work, it will be advantageous to optimize the network with many more metaheuristics such as cuckoo search technique, flower pollination algorithm, blood sugar regularization based optimization, ant colony optimization, firefly algorithm, chemical reaction optimization etc. to study the performance of the Fuzzy MLP net. It will also be interesting to hybridize the GSPSO model with one of aforesaid algorithm and to study the performance for real world pattern classification problems. GSPSO model can also be used to optimize higher order neural networks for these problems.

List of Publications

1. T. Dash, H.S. Behera, “A Fuzzy MLP Approach for Non-linear Pattern Classification,” *International Conference on Communication and Computing (ICC-2014)*, Publisher: **Elsevier**, Submission ID-134. ***(accepted, 02 April, 2014)*** (http://www.iccalpha.org)

2. T. Dash, H.S. Behera, “A Metaheuristic Fuzzy MLP Approach for Non-linear Classification and its Experimental Analysis,” under review in *Journal of Information Processing System (JIPS)* (Indexing: SCOPUS), Manuscript ID: 14E05-070.

3. T. Dash, H.S. Behera, “Hybrid Gravitational Search and Particle Swarm based Fuzzy MLP for Medical Data Classification,” submitted to *International Conference on Computational Intelligence in Data Mining (ICCIDM-2014)*, Publisher: **Springer’s Series on “Smart Innovation, Systems and Technologies”, Germany** (Indexing: SCOPUS). ISSN: 2190-3018. (http://www.iccidm.in)