Towards Crafting an Improved Functional Link Artificial Neural Network Based on Differential Evolution and Feature Selection

Ch. Sanjeev Kumar Dash and Ajit Kumar Behera
Silicon Institute of Technology, Silicon Hills
Patia, Bhubaneswar-751024, Odisha, India
E-mail: sanjeev_dash@yahoo.com, ajit_behera@hotmail.com

Satchidananda Dehuri
Department of Systems Engineering
Ajou University, San 5, Woncheon-dong
Yeongtong-gu, Suwon-443-749, South Korea
E-mail: satchi@ajou.ac.kr

Sung-Bae Cho
Soft Computing Laboratory
Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-gu
Seoul 120-749, South Korea
E-mail: sbcho@yonsei.ac.kr

Gi-Nam Wang
Department of Industrial Engineering
Ajou University, San 5, Woncheon-dong
Yeongtong-gu, Suwon-443-749, South Korea
E-mail: gnwang@ajou.ac.kr

Keywords: differential evolution, functional link artificial neural networks, classification, feature selection, genetic algorithms

Received: October 11, 2014

The proposed work describes an improved functional link artificial neural network (FLANN) for classification. The improvement in terms of classification accuracy of the network is realized through differential evolution (DE) and filter based feature selection approach. Information gain theory is used to filter out irrelevant features and provide relevant features to the functional expansion unit of FLANN as an input, which in turn maps low to high dimensional feature space for constructing an improved classifier. To fine tune the weight vector of the given network, differential evolution is used. The work is validated using skewed and balanced dataset retrieved from the University of California Irvine (UCI) repository. Our systematic experimental study divulges that the performance of the differential-evolution trained FLANN is promising than genetic algorithm trained FLANN, ISO-FLANN, and PSO-BP.

Povzetek: Predstavljena je gradnja klasifikacijske nevronske mreže, ki doseže boljše performanse z več novimi pristopi.

1 Introduction

Recently it is noticed that classification of big data [4] has demanded a great deal of attention. In this task, it is required to predict the value (the class label) of a user specified attribute based on the values of other predicting attributes. Although the task has been studied for many decades by the machine learning, statistics, and data mining communities but the complexity and shear size of the dataset creates lots of avenues in pursuit of perfection. Hence, an effort towards developing a smooth, accurate, and scalable classifier is always encouraging; it can face the challenge posed by the big data analysis. In this context, we urged that this work is a step towards handling big data, which has been plagued with many local optimal solutions and highly non-linear. Although, we have carried out our experimentation with the dataset obtained from the University of California, Irvine (UCI) repository [18] for validation, but it can be extended to handle big data.

Over the decades, neural networks [60] have been used as an alternative tool for approximating non-linearly separable boundary of classes in a classification problem. Pao et al. [41], have shown that FLANN may be conveniently used for function approximation and can be extended for classification with faster convergence rate and lesser computational load vis-à-vis multi-layer perceptron (MLP) structure. With this motivation, several classifiers such as adaptive Particle Swarm Optimization-Back-propagation (PSO-BP) learning [13], improved swarm optimized FLANN (ISO-FLANN) Dehuri, et al. [15] have already been developed with certain efficacy. The FLANN is basically a flat network and the need of the hidden layer is removed and hence the learning algorithm used in this network becomes very simple. The functional expansion effectively increases the dimensionality of the input vector and hence the hyper planes generated by the FLANN provide greater discrimination capability in the input pattern space.

Feature selection can be broadly classified into two categories: i) filter approach (it depends on generic statistical measurement); and ii) wrapper approach (based on the accuracy of a specific classifier) [2]. In the proposed work, the feature selection is performed based on information gain theory (entropy) measure with a goal to select a subset of features that preserves as much as possible the relevant information found in the entire set of features. We know that the architectural complexity of FLANN [12] is directly proportional to the number of features and the functions considered for expansion of the given feature value. Therefore, for reducing the architectural complexity, we first select a subset of features (i.e., feature selection) [28, 29] using gain ratio and then apply the usual procedure of function expansion and training by differential evolution [56]. In this work, the remarkable performance of DE as a global optimizer on continuous error function minimization problems has been studied [8] in the classification by effectively learning the FLANN. DE has also become a powerful tool for solving optimization problems that arise in other application areas like finance, medical, image processing [62], automatic clustering of big unlabeled datasets, et cetera.

This paper is set out as follows. Section 2 gives an overview of FLANN network, feature selection, and differential evolution. In Section 3, the proposed method is discussed. Experimental setup, results, and analysis are presented in Section 4. Section 5 concludes the paper with a future line of research

2 Background

The background of the research work is presented in this Section. In Subsections 2.1 and 2.2, literatures study of FLANN as a classifier and predictor is discussed. Feature selection and its importance are the focus of Subsection 2.3. Differential evolution, a meta-heuristic computing paradigm is discussed in Subsection 2.4.

2.1 Review of Literature

FLANNs are higher-order neural networks without hidden units introduced by Klasser and Pao [30]. Despite of their linear nature, FLANNs can capture non-linear input–output relationships, provided that these are fed with an adequate set of polynomial inputs, or the functions might be a subset of a complete set of orthonormal basis functions spanning through n-dimensional representation space, are constructed out of the original input attributes [41]. FLANNs can be used for non-linear prediction and classification. Related to this context, Subsections 2.1.1 and 2.1.2 are briefing out some of the works on FLANNs for classification and non-linear prediction.

2.1.1 FLANNs for Classification

In [53] a genetic algorithm used to select an appropriate number of polynomials as a functional input to the FLANN has been applied to the classification problem. However, their main concern was the selection of the optimal set of functional links to construct the classifier. In contrast, the proposed method gives much emphasis on how to develop the learning skill of the classifier by using filtered feature vectors. Misra and Dehuri [36] have used a FLANN for classification problem in data mining with a hope to get a compact classifier with less computational complexity and faster learning. Hu and Tseng [63] have used the functional link net known as BpFLANN for classification of bankruptcy prediction. With a motivation to restrict certain limitations, Dehuri, et al. [12] have coupled genetic algorithm based feature selection with FLANN (GFLANN). In the sequel, Dehuri and Cho [13] have given a road map on FLANN and designed a new PSO-BP adaptive learning mechanism for FLANN. In [14], Dehuri and Cho have contributed another stimulating work on FLANN [14] in succession with an improved swarm optimized FLANN for classification [15].

2.2 FLANNs for Prediction

Pao et al., have presented a functional link neural network (CoFLANN) in [40] to learn the control systems. They have shown several beneficial properties of generalized delta rule network with hidden layer and back-propagation (BP) learning. Haring and Kok [20], have proposed an algorithm (ClFLANN) using evolutionary computation (specifically genetic algorithm and genetic programming) for the determination of functional links (one based on polynomials and another based on expression tree) in neural network. Patra et al. [44] have proposed a CeFLANN based on BP learning and applied to the problem of channel equalization in a digital communication channel. Haring et al. [21], have proposed a ClaFLANN to select and transform features using evolutionary computation and showed that this kind of selection of features is a special case of so-called functional links. Hussain et al. [25] have described a new approach for decision feedback equalizer (DFE) based on the functional-link neural network (DfFLANN). The structure is applied to the problem of adaptive equalization in the presence of inter-symbol interference (ISI), additive white Gaussian noise, and co-channel interference (CCI). The experimental results provide significantly superior bit-error rate (BER) performance characteristics as compared to the conventional methods. Chen et al. [6] have presented an adaptive implementation of the functional-link neural network

(AFLNN) architecture together with a supervised learning algorithm named Rank-Expansion with Instant Learning (REIL) that rapidly determines the weights of the network. The beauty of their proposed algorithm is one-shot training as opposed to iterative training algorithms in the literature. Dash et al. [9], have proposed an ElfFLANN with trigonometric basis functions to forecast the short-term electric load. Panagiotopoulos et al. [39] have reported better results by applying FLANN for planning in an interactive environment between two systems: the challenger and the responder. Patra et al. [44] have proposed a FLANN with BP learning (SiFLANN) for identification of non-linear dynamic systems. Moreover, Patra et al. [44] have used FLANN to adaptive channel equalization in a digital communication system with 4-QAM signal constellation named as QsFLANN. They have compared the performance of the FLANN with a multilayer perceptron (MLP) and a polynomial perceptron network (PPN) along with a conventional linear LMS-based equalizer for different linear and nonlinear channel models. Out of the three ANN equalizer structures, the performance of the FLANN is found to be the best in terms of MSE level, convergence rate, BER and computational complexity for linear as well as nonlinear channel models over a wide range of SNR and EVR variations. With the encouraging performance of FLANN [47, 48, 49], Patra et al. [45] have further motivated and came up with another FLANN known as IpFLANN with three sets of basis functions such as Chebyshev, Legendre, and power series to develop an intelligent model of the CPS involving less computational complexity. In the sequel, its implementation can be economical and robust. Park and Pao [43] have reported the performance of a holistic-styled word-based approach to off-line recognition of English language script. The authors have combined the practices of radial basis function neural net (RBNN) and the random vector functional-link net approach (RVFLANN) and obtained a method called the density-based random-vector functional-link net (DBRVFLANN). The combination is helpful in improving the performance of word recognition. A Chebyshev functional link artificial neural networks (CFLANN) is proposed by Patra et al. [49] for non-linear dynamic system identification. Sing et al. [54] has estimated the degree of insecurity in a power system by the proposed IeFLANN with a set of orthonormal trigonometric basis functions. An evolutionary search of genetic type and multi-objective optimization [34] such as accuracy and complexity of the FLANN in the Pareto sense is used to design a generalized FLANN (SyFLANN) with internal dynamics and applied to system identification. A reduced-decision feedback functional link artificial neural network (RDF-FLANN) structure for the design of a nonlinear channel equalizer in digital communication systems is proposed by Weng et al. [58]. Authors have reported that the use of direct decision feedback can greatly improve the performance of FLANN structures. Weng et al. [57], have proposed a reduced decision feed-back Chebyshev functional link artificial neural networks (RDF-CFLANN) for channel equalization. In [46], FLANNs with trigonometric polynomial functions (IsFLNN) are used in intelligent sensors for harsh environment that effectively linearizes the response characteristics, compensates for non-idealises and calibrates automatically.

Interval regression analysis has been a useful tool for dealing with uncertain and imprecise data. Since the available data often contain outliers, robust methods for interval regression analysis are necessary. Hu [24] has proposed a genetic-algorithm-based method (IraFLANN) for determining two functional-link nets for the robust nonlinear interval regression model: one for identifying the upper bound of data interval, and the other for identifying the lower bound of data interval.

2.3 FLANNs Classifier

The FLANN architecture [10, 47, 48, 49] uses a single layer feed forward neural network by removing the concept of hidden layers. The learning of a FLANN may be considered as approximating or interpolating a continuous, multivariate function *f*(X) by an approximating function $f_W(X)$. In FLANN a set of basis functions Φ and a fixed number of weight parameters W are used to represent $f_W(X)$. With a specific choice of a set of basis functions, the problem is to find the weight parameters W that provides the best possible approximation of 'f' on the set of input-output examples. So, the most important thing is that how to choose the basis functions to obtain better approximation.

Let us consider a set of basis function $\gamma = \{\Phi_i \in L(A)\}_{i \in I}$ with the following properties:

(i) Φ_1=1, (ii) the subset $\gamma_j = \{\Phi_i \in \gamma\}_{i=1}^{j}$ is a linearly independent set, i.e., if $\sum_{i=1}^{j}(w_i \Phi_i) = 0$, then $w_i = 0$ for all *i*=1,2,…*j*, and

(iii) $\sup_j \left[\sum_{i=1}^{j} \|\Phi_i\|_A^2 \right]^{1/2} < \infty$.

Let $\gamma_N = \{\Phi\}_{i=1}^{N}$ be a set of basis functions to be considered for FLANN. Thus, the FLANN consists of *N* basis functions $\{\Phi_1, \Phi_2, ..., \Phi_N\} \in \gamma_N$ with the following input-output relationship for the jth output:

$$\hat{y} = \rho(s_j); \; s_j = \sum_{i=1}^{N} (w_{ji}.\Phi_i(X)) , \quad (1)$$

where $X \in A \subset R^n$, i.e., $X = [x_1, x_2, ..., x_n]^T$ is the input pattern vector, $\hat{y} \in R^m$ i.e., $\hat{y} = [y_1, y_2, ..., y_m]^T$ is the output vector and $w_j = [w_{j1}, w_{j2}, ..., w_{jN}]$ is the weight vector associated with the jth output of the FLANN. The non-linear function $\rho(.) = \tanh(.)$ is used to transfer the weighted sum into desired output format of an input pattern.

Considering the *m*-dimension output vector, equation (1) can be written in matrix notation as follows:

$$S = W\Phi, \tag{2}$$

where W is an $(m \times N)$ weight matrix of the FLANN given by $W = [w_1, w_2, ..., w_m]^T$, $\Phi = [\Phi_1(X), \Phi_2(X), ..., \Phi_N(X)]^T$ is the basis function vector, and $S = [S_1, S_2, ..., S_N]^T$ is a matrix of linear outputs of the FLANN. The *m*-dimensional output vector $\hat{y}$ may be given by

$$\hat{y} = \rho(s) = f_w(X), \tag{3}$$

The training of the network is done in following way:

Let 'k' patterns be applied to the network in a sequence repeatedly. Let the training sequence be denoted by (X_k, y_k) and the weight of the network be $W(k)$, where the 'k' is also the iteration. Referring to equation (1) the *j*th output of the FLANN at iteration *k* is given by:

$$\hat{y}(k) = \rho\left(\sum_{i=1}^{N} \left(w_{ji}(k)\Phi_i(X_k)\right)\right) = \rho(w_j(k)\Phi^T(X_k)), \tag{4}$$

for all $X \in A$ and *j*=1,2…,*m*, where $\Phi(X_k) = [\Phi_1(X_1), \Phi_2(X_2), ..., \Phi_N(X_k)]^T$.

Let the corresponding error be denoted by:

$$e_j(k) = y_j(k) - \hat{y}_j(k).$$

In words, the weighted sum of the functionally expanded features is fed to the single neuron of the output layer of the FLANN. The weights are optimized by the DE method during the process of training. The set of functions considered for function expansion may not be always suitable for mapping the non-linearity of the complex task. In such cases few more functions may be incorporated into the set of functions considered for expansion of the input data set.

However, dimensionality of many problems itself is very high and further increasing the dimensionality to a very large extent may not be an appropriate choice. So, this is one of the reasons, why we are carrying out this work.

2.4 Feature Selection

Feature selection (FS) [29] is an essential task to remove irrelevant and/or redundant features. In other words, feature selection techniques provide a way to select a subset of potential attributes or variables from a dataset. For a given classification problem, the network may become unbelievably complex if the number of the features used to classify the pattern increases very much. So the reason behind using FS techniques include reducing dimensionality by removing irrelevant and redundant features, reducing the amount of attributes needed for learning, improving algorithms' predictive accuracy, and increasing the constructed model's comprehensibility. After feature selection a subset of the original features is obtained which retains sufficient information to discriminate well among classes. The selection of features can be achieved in two ways:

Filter Method: The filter approach is independent of the learning algorithm, computationally simple, fast, and scalable. Using filter method, feature selection is done once and then can be provided as input to different classifiers. In this method features are ranked according to some criterion and the top k features are selected.

Wrapper model: This approach uses the method of classification itself to measure the importance of feature sets; hence the selected features depend on the classifier model used [26]. In this method a minimum subset of features is selected without learning performance deterioration.

Wrapper methods generally result in better performance than filter methods because the feature selection process is optimized for the classification algorithm to be used. However, wrapper methods are too expensive for large dimensional database in terms of computational complexity and time since each feature set considered must be evaluated with the classifier algorithm used. Filter based feature selection methods are in general faster than wrapper based methods.

2.5 Differential Evolution

Differential evolution (DE) [50, 55, 57] is a population based stochastic search algorithm. As a stochastic optimizer, it has the capability to handle non-linearity, non-convexity, multi-modality, and even dynamic characteristics of the problem. Unlike canonical GA, it typically operates on real valued individual encodings. Like GAs [35], DE maintains a pool of potential solutions which are then perturbed in an effort to explore yet better solutions to a problem in hand. In GAs, the individuals are perturbed based on *crossover* and *mutation*. However in DE, individuals are perturbed based on the difference of different individuals, borrowing ideas from the Nelder-Mead simplex method [50]. One of the advantages of this approach is that the resulting 'step' size and orientation during the perturbation process automatically adapts to the landscape of fitness function.

There are many variants of DE algorithms developed [8, 50] in past few years, the most classical variants is based on the *DE/rand/1/bin* scheme [55]. The different variants of the DE algorithm are described using the notation *DE/x/y/z*, where *x* specifies how the base vector is chosen (e.g., *rand*-if it is randomly selected, or *best*-if the best individual is selected), *y* is the number of difference vectors used, and *z* denotes the crossover scheme (*bin* for crossover based on independent binomial experiments, and *exp* for exponential crossover).

A pool of *n*, *d*-dimensional solution vectors $x_i = (x_{i1}, x_{i2}, ..., x_{id}), i = 1, 2, ..., n$ is randomly initialized and evaluated using a fitness function *f*(.). During the process of search, each individual (*i*) is iteratively refined. The following three steps are iterated one after another till desired optimum is reached.

i) **Mutation**: Create a donor vector which encodes a solution, using randomly selected members of the population.

ii) Crossover: Create a trial vector by combining the donor vector with *i*.

iii) Selection: By the process of selection, determine whether the newly-created trial vector replaces *i* in the population or not.

The pseudo-code of DE is illustrated as follows:

```
DE (Scaled Factor $f_m$, Crossover Rate $C_r$, Pool Size $n$)
{
  INITIALIZATION: Generate a population of n, d-dimensional solution vectors in the search space.
  DO
    FOR each i of the Population of individuals
        MUTATION: Generate a donor vector $v_i$ using equation (5).
        CROSSOVER: Generate a trial vector $u_i$ using equation (6).
        SELECTION: Evaluate the trial vector ($u_i$). if $f(u_i) > f(x_i)$ (for maximization problem) then replace $x_i$ by $u_i$ else $x_i$ will survive to next generation.
    END FOR
    WHILE (Termination Criterion Met)
    }
```

For the mutation of *i*th individual of the population, three different individuals' x_{r1}, x_{r2}, and x_{r3} with $r1 \neq r2 \neq r3 \neq i$ will be randomly chosen from the pool to generate a new vector known as *donor* vector. The donor vector is described as follows.

$$v_i = \underbrace{x_{r1}}_{\substack{base\\Vector}} + f_m.\underbrace{(x_{r2} - x_{r3})}_{\substack{Individial\\Difference}}, \qquad (5)$$

where, the scaling parameter f_m called mutation factor and a general setting for the parameter is $f_m \in [0,2]$. However, Storn and Price suggest $f_m \in [0.5,1]$ as such a setting may result in good optimization effectiveness.

Selecting three indices randomly imply that all individuals of the current pool have the same chance of being selected, and therefore influencing the creation of the difference vector. The mutation factor controls the amplification of the difference vector and in turn used to avoid stagnation of the search process. There are several alternative versions of the above process for creating a donor vector [for details see [8, 50]. In [62] a self-adaptive DE is used that can tune this scaling factor dynamically.

After the creation of the donor vector (v_i), a binomial crossover (*bin*) operates on the vector v_i and the target vector x_i to generate a trial vector in the following way.

$$u_{ij} = \begin{cases} v_{ij} & if\,(rand \le c_r)\ or\ \ (j = rand(1,2,...,d)) \\ x_{ij} & if\,(rand > c_r)\, and\, (j \ne rand(1,2,....,d)) \end{cases}, \qquad (6)$$

where x_{ij}, v_{ij}, and u_{ij} are the *j*-dimensional components of the vectors x_i, v_i, and u_i, respectively; *rand* is a random number generated in the range (0, 1); is the user-specified crossover constant from the range (0, 1). The resulting trial (child) vector replaces its parent if it has higher fitness (a form of selection); otherwise the parent survives unchanged into the next iteration of the algorithm (shown in equation (7)).

$$x_i(t+1) = \begin{cases} u_i(t) & if\,(f(u_i(t)) > f(x_i(t))) \\ x_i(t) & otherwise \end{cases} \qquad (7)$$

It is provided a comprehensive comparison of the performance of DE with a range of other optimizers, including GA, and report that the results obtained by DE are consistently as good as the best obtained by other optimizers across a wide range of problem instances in [50]. There are a number of reasons to integrate FLANN with DE. The first reason is to reduce local optimality during the training of FLANN. Although genetic algorithm coupled with FLANN has already been established to reduce local optimality while designing a classifier but DE has some merits over GA. Therefore, we are ignited to carry out this work. Secondly, the real encoding of DE solves the problem of encoding-decoding mapping. Thirdly, it can achieve faster convergence speed. Lastly DE does not undergo any complex process of parameter tuning and works very reliably with excellent overall results.

3 Proposed Method

With an objective to design a smooth and accurate classifier, the proposed approach is combining the idea of filter based feature selection and simple FLANN classifier [15]. It is a two phase method. In phase one, we are selecting a set of relevant features by using the entropy while in the second phase the weights of FLANN are trained using differential evolution. Figure 1 depicts the overall architecture of the approach.

In the first phase, we rank the features or attributes according to information gain ratio and then delete an

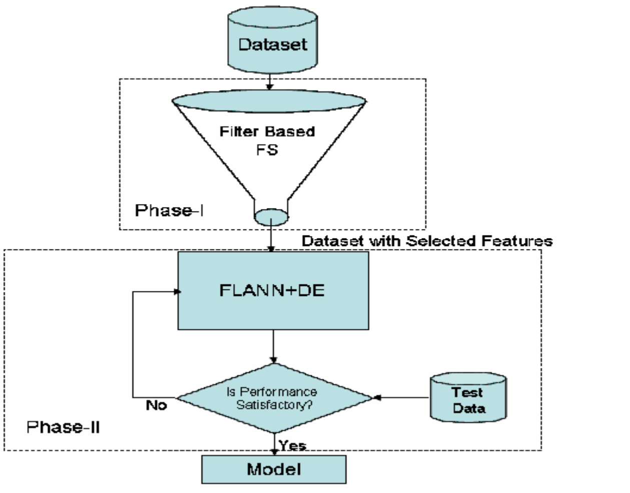

Figure 1: Architecture of Proposed Method.

appropriate number of features which have the least gain ratio [2]. The exact number of features deleted varies from dataset to dataset. The expected information needed to classify a tuple in D is given by equation (8),

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i), \quad (8)$$

where p_i is the non-zero probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_{i,D}|/|D|$. A log function to the base 2 is used, because the information is encoded in bits. *Info*(*D*) is the average amount of information needed to identify the class level of a tuple in D. *Info*(*D*) is also known as entropy of *D* and is based upon only the properties of classes.
For an attribute 'A' entropy "$Info_A(D)$" is the information still required to classify the tuples in D after partitioning tuples in D into groups only on its basis.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j), \quad (9)$$

where *v* is the number of distinct values in the attribute A, |*D*| is the total number of tuples in *D* and $|D_j|$ is the number of repetitions of the j^{th} distinct value.
Information gain (*Gain*(*A*)) is defined as the difference between the original information requirement and new requirement (after partitioning on A) (refer equation 10)

$$Gain(A) = Info(D) - Info_A(D). \quad (10)$$

Information gain applies a kind of normalization to information gain using split information value defined analogously with $Info(D)$ as equation 11:

$$SplitInfo_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right). \quad (11)$$

This value represents the potential information generated by splitting the training data set, D, into *v* partitions, corresponding to the *v* outcomes of test on attribute A. For each outcome, it considers the number of tuples having the outcome with respect to the total number of tuples in D. The gain ratio is defined as in equation (12).

$$GainRatio(A) = Gain(A) / SplitInfo(A). \quad (12)$$

In summary, the feature selection is done in first phase using information gain ratio and then the dataset with reduced number of features is used for automatic training and determination of the parameters of FLANN using DE in the second phase.

In the second phase, we are focusing on the learning of the classifier. Here, differential evolution is employed to reveal the weight of the FLANN. This ensures efficient representation of an individual of DE. Since the performances of the FLANN mainly depend on weight; we just encode the weight into an individual for stochastic search. We have chosen a set of trigonometric functions as the basis function for functional expansion. The reason of choosing trigonometric functions for functional expansion is as follows:

Without loss of generality, for all the polynomials of Nth order with respect to an orthonormal system $\{\varphi_i(u)\}_{i=1}^{N}$ the best approximation in the metric space L^2 is given by the Nth partial sum of its Fourier series with respect to the system. Thus, the trigonometric polynomial basis functions provide a compact representation of the function in the mean square sense. However, when the outer product terms were used along with the trigonometric polynomials for functional expansion, better results were obtained in the case of learning the classifier.

Suppose the maximum number of trigonometric functions used to expand a particular feature is '*F*' and there are 'L' features selected for input to the network, then the size of the weight vector is defined as $K_{\max} = L.(F+1)$, then the length of the individual is $K_{\max} + B$. The structure of the individual is represented in Figure 2.

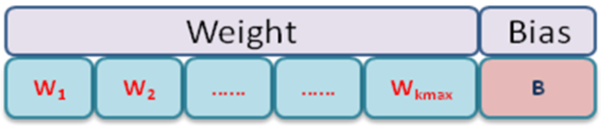

Figure 2: Structure of the Individual.

In other words, each individual has two constituent parts such as weight and bias.

The fitness function which is used to guide the search process is defined in equation (13).

$$E = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e(i)^2}, \quad e(i) = y(i) - \hat{y}(i), \quad (13)$$

where N is the total number of training sample $y(i)$ is the actual output of ith pattern and $\hat{y}(i)$ is the estimated output of FLANN. The error and hence root mean square is denoted as e(i) and E respectively.

The algorithmic framework of FLANN-DE is described as follows:

Initially, a set of n_p individuals (i.e., *i*=1,2,..,n_p) is the size of the population) pertaining to networks weights and bias are created.

$$x_i^{(t)} = \left\langle x_{i1}^{(t)}, x_{i2}^{(t)}, \ldots, x_{id}^{(t)} \right\rangle, \quad i = 1, 2, \ldots, n_p$$

where $d = K_{\max} + B$ and *t* is the iteration number.

At the start of the algorithm this n_p set of individuals is initialized randomly and then evaluated using the fitness function *f* (.).

In each iteration, e.g., iteration *t*, for individual $x_i^{(t)}$ undergoes mutation, crossover, and selection as follows:
Mutation: for vector $x_i^{(t)}$ a perturbed vector $V_i^{(t+1)}$ called donor vector is generated according to equation (14):

$$V_i^{(t+1)} = x_{r1}^{(t)} + m_f \times (x_{r2}^{(t)} - x_{r3}^{(t)}), \quad (14)$$

where m_f is the mutation factor drawn from (0,2], the indices r_1, r_2, and r_3 are selected randomly from $\{1,2,3,\ldots,n_p\}$.

Crossover: The trial vector is generated as follows (equation (15)):

$$u_i^{(t+1)} = \left\langle u_{i1}^{(t+1)}, u_{i2}^{(t+1)}, \ldots, u_{id}^{(t+1)} \right\rangle,$$

$$u_{ij}^{(t+1)} = \begin{cases} v_{ij}^{(t+1)} & if\ (rand \le c_r) \quad or\ \ (i = rand(1,2,..,d)) \\ x_{ij}^{(t)} & if\ (rand > c_r) \quad and\ (i \ne rand(1,2..,d)) \end{cases}, \quad (15)$$

where j=1, 2, …,d, *rand* is a random number generated in the range (0,1) c_r is the user specified crossover constant from the range (0,1) and *rand*(1,2,…,*d*) is a randomly chosen index from the range (1,2,…,d). The random index is used to ensure that the trial solution vector differs by at least one element from $x_i^{(t)}$. The resulting trial (child) solution replaces its parent if it has higher accuracy (a form of selection), otherwise the parent survives unchanged into the next iteration of the algorithm.

Finally, we use selection operation and obtain target vector $x_i^{(t+1)}$ as follows in equation (16):

$$x_i^{(t+1)} = \begin{cases} u_i^{(t+1)} & if \quad f(x_i^{(t+1)}) \le f(x_i^{(t)}) \\ x_i^{(t)} & otherwise \end{cases}. \quad (16)$$

4 Experimental Study

The data set and experimental parameter setting are discussed in Subsection 4.1. Results are analyzed in Subsection 4.2.

4.1 Description of Dataset Parameters

The data set used to test the proposed method obtained from the UCI machine learning repository [18]. Four balanced and unbalanced datasets have been chosen to validate the proposed method. The details about the four data sets are given below.

Iris: This data set includes 150 instances and each having 4 attributes, excluding the class attribute. The instances are uniformly classified into 3 classes (i.e., every instance either belongs to class 1, or 2, or 3). Class 1 has 50 instances, class 2 contains 50, and remaining instances (i.e., 50) are belong to class 3. None of the attributes contain any missing values. All attributes are continuous.

Wine: This data set includes 178 instances and each having 13 attributes, excluding the class attribute. The instances are classified into 3 classes (i.e., every instance either belongs to class 1 or 2 or 3) in an almost balanced way. Class 1 has 59 instances, class 2 contains 71, and remaining instances (i.e., 48) are belong to class 3. None of the attributes contain any missing values. All attributes are continuous.

Lymphography: This data set includes 148 instances and each having 19 attributes including the class attribute. The instances are classified into 4 classes (i.e., every instance either belongs to class 1, or 2, or 3, or 4). Class 1 has 2 instances, class 2 contains 81, class 3 contains 61 and remaining instances (i.e., 4) belong to class 4. None of the attributes contain any missing values. All attributes are continuous. However, this dataset is purely unbalanced.

Stalog(Heart): There are 270 instances, 13 attributes, and 2 classes in this dataset. Class 1 has 151 instances and Class 2 has 119 instances. None of the attributes contain any missing values. The distributions of samples into different classes are almost balanced.

Pima: This data set includes 768 instances and each having 8 attributes along with one class attribute. The instances are classified into 2 classes (i.e., every instance either belongs to class 1 or 2). Class 1 has 500 instances and class 2 contains 268. None of the attributes contain any missing values. All attributes are continuous. However, the distribution of samples into various classes is not balanced.

In our experiment, every dataset is randomly divided into two mutually exclusive parts: 50% as training sets and 50% as test sets. The parameters' value used for validating our proposed method is listed in Table 1. These parameters were obtained after several rounds of independent runs. However, the number of iterations are varies from dataset to dataset.

Table 1: Parameters used for simulation

Parameter		**Iris**	**Wine**	**Lympho - graphy**	**Stalog (heart)**	**Pima**
Population		50	50	50	50	50
Mutation		0.2	0.4	0.3	0.4	0.4
Crossover		0.8	0.8	0.6	0.8	0.8

In the case of Iris dataset, an ideal number of iterations is lies within the range of 40~50. However, it varies from 400~500 in the case of Lymphography, Statlog (Heart), and Pima. In the case of Wine the ideal number of iteration can varies from 1000~1200. Similarly, the parameters setting of the methods ISO-FLANN and PSO-BP have been fixed as suggested in the respective literatures.

4.2 Result Analysis

The experimental test results are presented in Table 2. The accuracy in terms of percentage of test samples correctly classified using proposed method and method ISO-FLANN, PSO-BP, and method proposed in [12] are given in columns 2, 3, 4, and 5 of Table 2. In all the cases, 1/3rd of the features have been removed.

In our comparison, it is noticed that the accuracy obtained from our proposed method is better than the method proposed in [12], ISO-FLANN, and PSO-BP, moreover a paired t-test has been performed to judge the performance all the algorithms properly. With the 5% significance level, the critical value of t is 0.43 and it is not coming under the specified range, so the null hypothesis is rejected.

Table 2: Testing accuracy of proposed method vs. method proposed in ISO-FLANN (Dehuri, et al., 2012), PSO-BP (Zhang, et al., 2007) and (Dehuri, Mishra, and Cho, 2008).

Dataset	**Proposed Method**	**ISO-FLANN**	**PSO-BP**	**Method Proposed in (**Dehuri, Mishra, and Cho, 2008**)**
Iris	98.33	97.62	97.12	97.33
Wine	93.10	92.32	91.54	90.45
Lymphography	87.50	86.9	85.65	77.08
Statlog (heart)	86.57	85.46	84.77	84.45
Pima	79.20	78.86	76.88	72.14

The features which are identified to remove during the filter process are listed in Table 3. Further, the mapping of numeric and actual name of the features are shown in Table 4 to avoid confusion or ordering.

Table 3: Filtered Attributes of Datasets.

Dataset	**Attributes of the Dataset**	**Attributes Removed**
Iris	1~4	2
Wine	1~13	3,4,5,8
Lymphography	1~18	1,6,9,12,16,17
Statlog(heart)	1~13	1,4,6,7
Pima	1~8	1,3

Table 4: Features ordering of all datasets.

Dataset	**Attributes Ordering**	**Name**
Iris	1	Sepal Length
	2	Sepal Width
	3	Petal Length
	4	Petal Width
Wine	1	Alcohol
	2	Malic acid
	3	Ash
	4	Alcalinity of ash
	5	Magnesium
	6	Total phenols
	7	Flavanoids
	8	Nonflavanoid phenols
	9	Proanthocyanins
	10	Color intensity
	11	Hue
	12	OD280/OD315 of diluted wines
	13	Proline
Lymphography	1	Lymphatics: normal, arched, deformed, displaced
	2	Block of affere: no, yes
	3	Bl. of lymph. c: no, yes
	4	Bl. of lymph. s: no, yes
	5	By pass: no, yes
	6	Extravasates: no, yes
	7	Regeneration of: no, yes
	8	Early uptake in: no, yes
	9	Lym.nodes dimin: 0-3
	10	Lym.nodes enlar: 1-4
	11	Changes in lym.: bean, oval, round
	12	Defect in node: no, lacunar, lac. marginal, lac. central
	13	Changes in node: no, lacunar, lac. margin, lac. central
	14	Changes in stru: no, grainy, drop-like, coarse, diluted, reticular, stripped, faint
	15	Special forms: no, chalices, vesicles
	16	Dislocation of: no, yes
	17	Exclusion of no: no, yes
	18	No. of nodes in: 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, >=70
Statlog (heart)	1	Age
	2	Sex
	3	Chest pain type (4 values)
	4	Resting blood pressure
	5	Serum cholestoral in mg/dl
	6	Fasting blood sugar > 120 mg/dl
	7	Resting electrocardiographic results (values 0,1,2)
	8	Maximum heart rate achieved
	9	Exercise induced angina
	10	Oldpeak = ST depression induced by exercise relative to rest
	11	The slope of the peak exercise ST segment
	12	Number of major

		vessels (0-3) colored by flourosopy
	13	Thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
Pima	1	Number of times pregnant
	2	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
	3	Diastolic blood pressure (mm Hg)
	4	Triceps skin fold thickness (mm)
	5	2-Hour serum insulin (mu U/ml)
	6	Body mass index (weight in kg/(height in m)^2)
	7	Diabetes pedigree function
	8	Age (years)

The error rate obtained from the proposed method varies over a number of iterations of IRIS, Lymphography, WINE, Statlog (Heart), and PIMA are illustrated in Figures 3, 4, 5, 6, and 7.

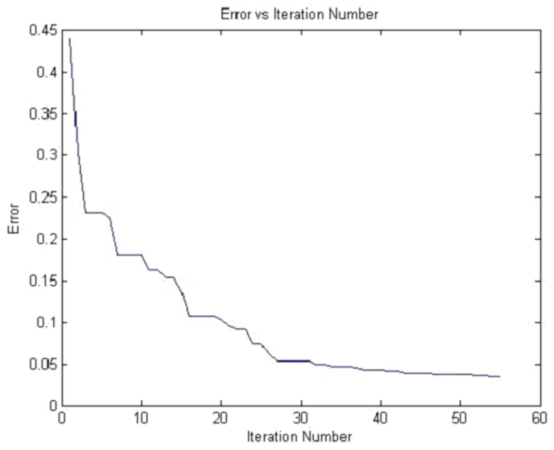

Figure 3. Iteration Number vs Error Obtained from IRIS Dataset.

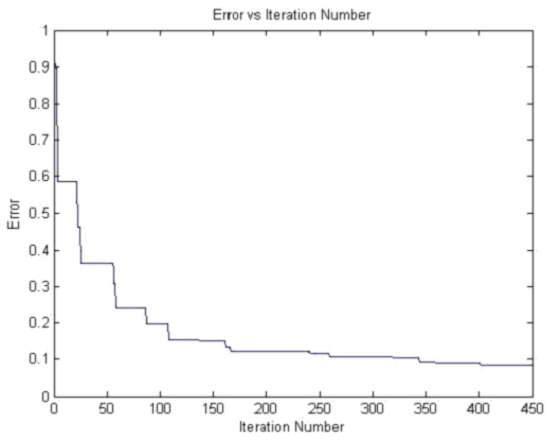

Figure 4. Iteration Number vs. Error Obtained from Lymphography Dataset.

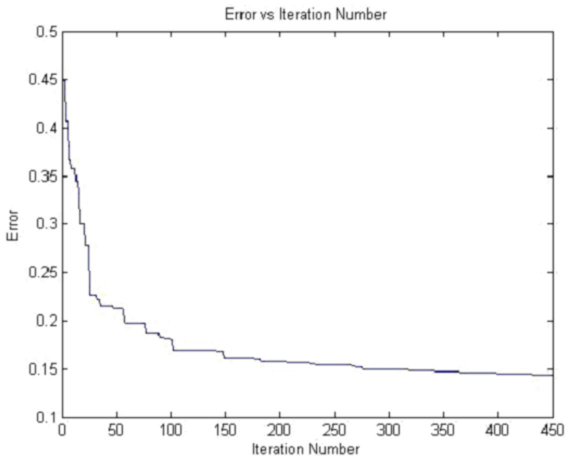

Figure 5. Iteration Number vs. Error Obtained from PIMA Dataset.

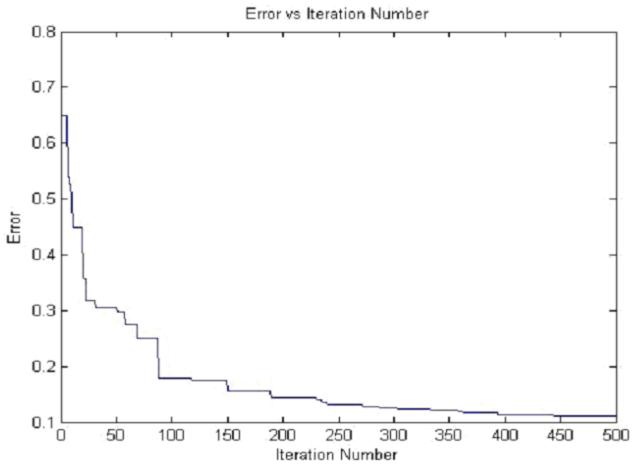

Figure 6. Iteration Number vs Error Obtained from Statlog (Heart) Dataset.

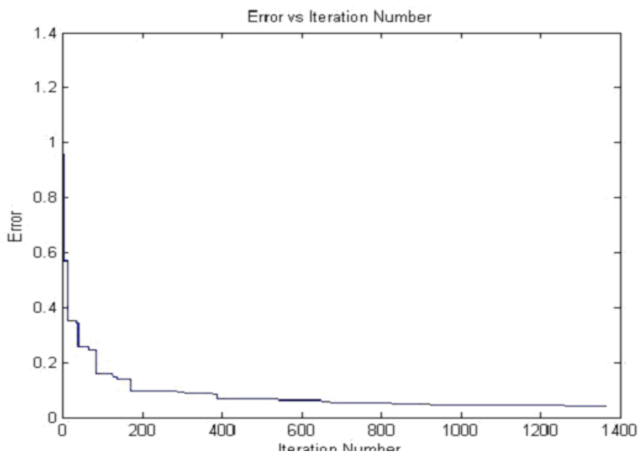

Figure 7. Iteration Number vs. Error Obtained from WINE Dataset.

Further, it is interestingly noticed that if we do not eliminate any features from the dataset then the accuracy of the results hardly makes any difference, except wine dataset. In the case of wine dataset the accuracy obtained with feature selection is poor than without feature selection. With respect to optimal mutation factor the results of proposed method without feature selection for different datasets are described in Table 5. However, it is better than ISO-FLANN, PSO-BP, and method proposed in [12].

Name of Data set	Mutati on factor	Accuracy in percentage
Iris	0.3	98.23
Wine	0.4	98.51
Lymphography	0.3	87.50
Stalog(heart)	0.4	86.56
Pima	0.4	78.91

Table 5: Results Obtained from proposed method without feature selection.

Figures 8, 9, 10, 11, and 12 illustrate the performance of the proposed method without feature selection over different mutation rate. In the case of Iris and Lymphography the accuracy of the proposed classifier dropped after the mutation rate 0.3. Similarly, in the case of Wine, Pima, and Statlog the accuracy dropped after the mutation rate 0.4. Hence, our experimental study recommends the mutation rate setup mentioned in Table 5.

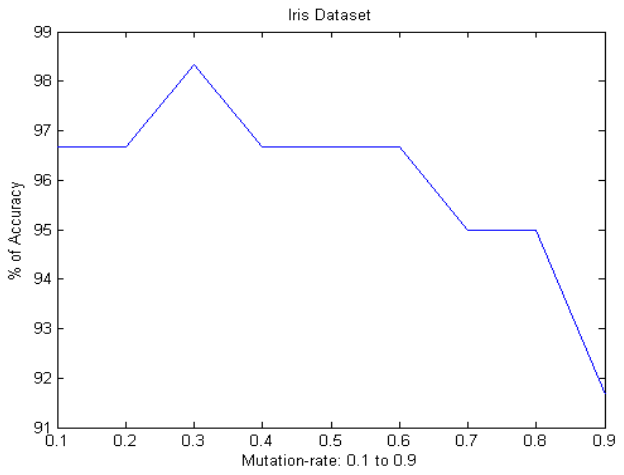

Figure 8: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from IRIS.

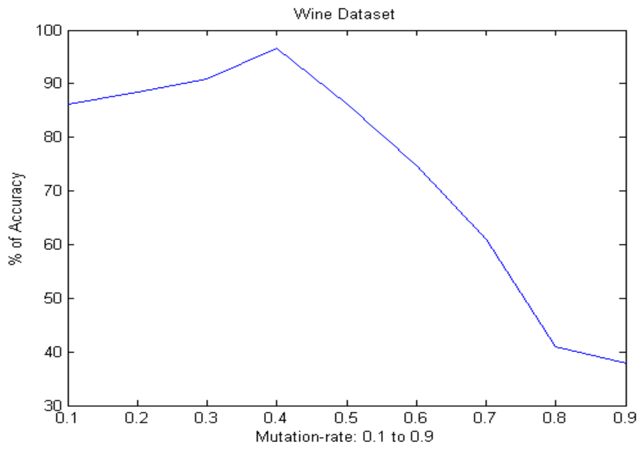

Figure 9: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from WINE.

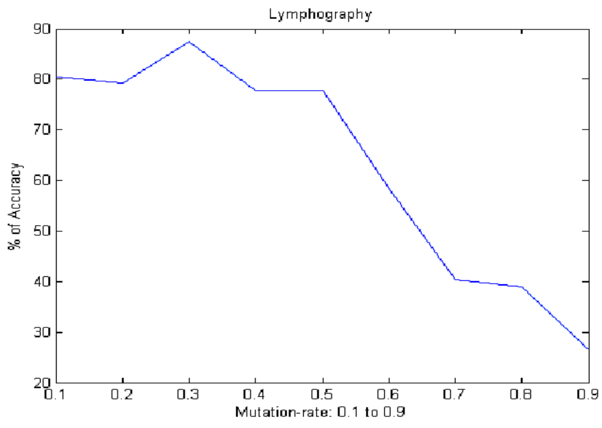

Figure 10: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from Lymhography.

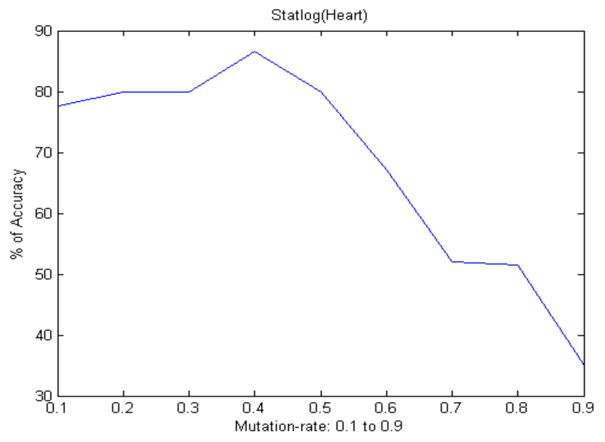

Figure 11: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from Statlog (Heart).

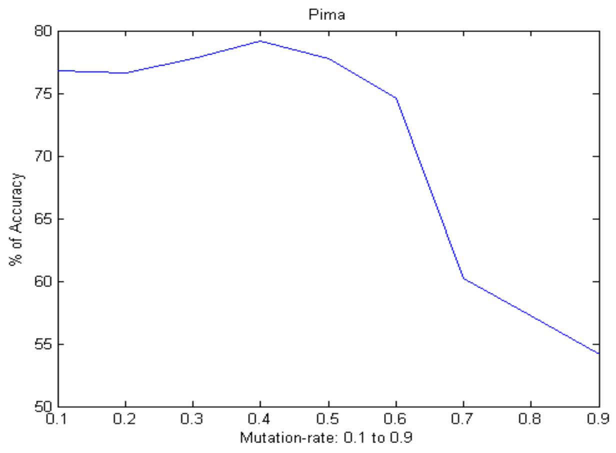

Figure 12: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from PIMA.

Figures 13, 14, 15, 16, and 17 demonstrate; how accuracy of the proposed classifier with filter based feature selection varies over different mutation rates.

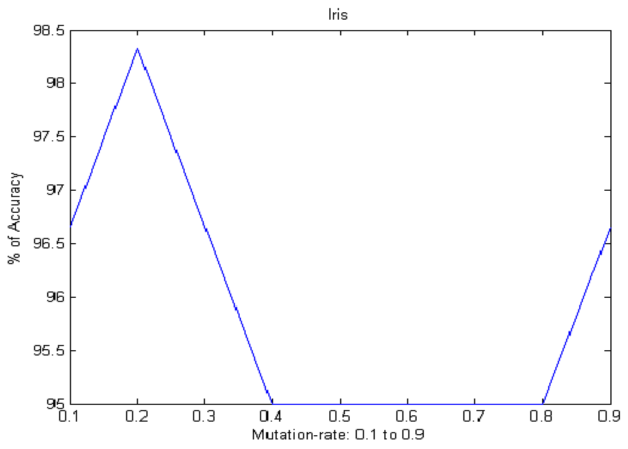

Figure 13. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) Obtained from IRIS.

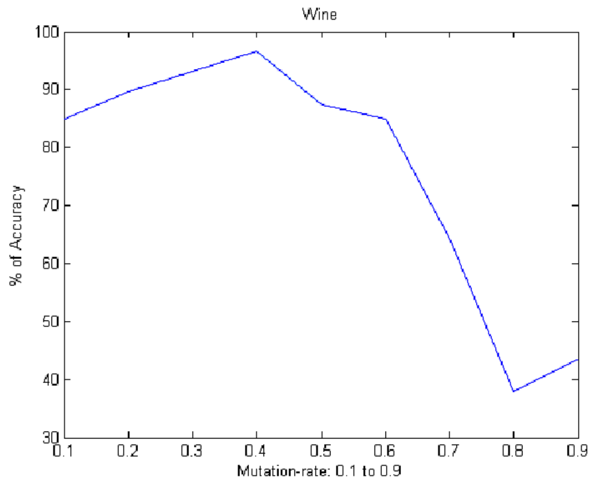

Figure 14. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from WINE.

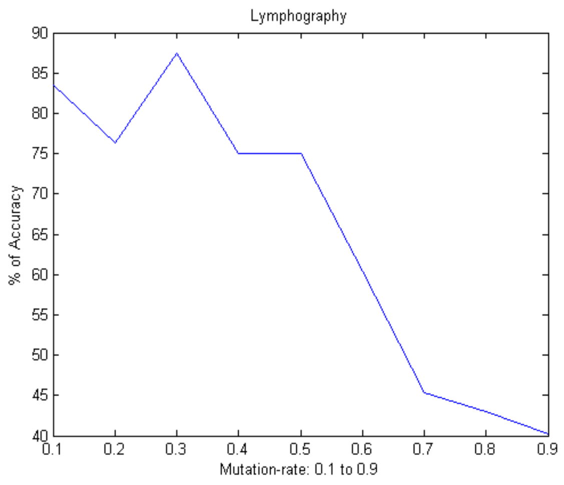

Figure 15. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from Lympography.

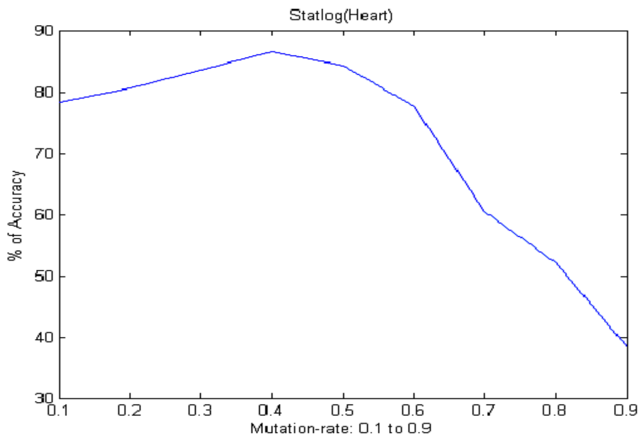

Figure 16. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from Statlog (heart).

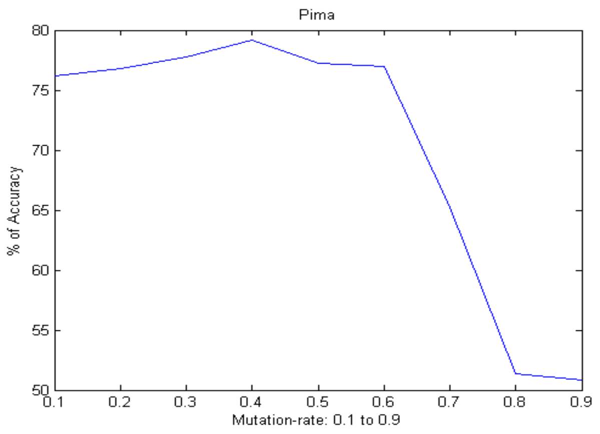

Figure 17. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from PIMA.

In the case of Iris the accuracy of the proposed classifier with mutation rate 0.2 is promising. In the case of Wine and Lymphography the accuracy is dropped after the mutation rate 0.3. Similarly the accuracy of Statlog and Pima has been dropped after the mutation rate 0.4.

5 Conclusion

An integrated framework of differential evolution and FLANN along with a filter based feature selection has been crafted in this work to classify unknown patterns. The experimental results confirm that the combination of filter based feature selection and training of FLANN using differential evolution obtains accurate and smooth classification. Further, it has been observed that if we reduce 1/3rd of total attributes by the process of filter approach, then the network shows an improvement and constancy in accuracy. Finally, we have compared the experimental outcomes obtained from this study with its rival proposed by Dehuri et al. [12], ISO-FLANN, and PSO-BP, we then conclude that the accuracy draws a very sharp edge between the proposed method and the rest of the methods consider for comparison. Our future line of research includes: i) validation of its accuracy and scalability in big data analysis and ii) dealing with noise and uncertainty of the dataset by suitable integration with other soft computing paradigm.

References

[1] Abu-Mahfouz, I.-A. (2005). A Comparative Study of Three Artificial Neural Networks for the Detection and Classification of Gear Faults. *International Journal of General Systems*, 34(3), pp. 261–277.

[2] Aruna, S., Nandakishore, L. V., and Rajagopalan, S. P. (2012). A Hybrid Feature Selection Method based on IGSBFS and Naïve Bayes for the Diagnosis of Erythemato-Squamous Diseases. *International Journal of Computer Applications* (0975-8887), 41(7).

[3] Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4), 537-550.

[4] Bifet, A. (2013). Mining Big Data in Real Time. *Informatica (Slovenia)*, 37(1), 15-20.

[5] Carvalho, D. R., & Freitas, A. A. (2004). A hybrid decision tree/genetic algorithm method for data mining. *Information Sciences*, 163(1), 13-35.

[6] Chen, C. P., LeClair, S. R., & Pao, Y. H. (1998). An incremental adaptive implementation of functional-link processing for function approximation, time-series prediction, and system identification. *Neurocomputing*, 18(1), 11-31.

[7] Das, M., Roy, R., Dehuri, S., and Cho, S.-B. (2011). A New Approach to Associative Classification Based on Binary Multi-objective Particle Swarm Optimization. *International Journal of Applied Meta-heuristic Computing*, 2(2), pp. 51-73.

[8] Das, S. and Suganthan, P. N. (2011). Differential Evolution: A Survey of the State-of- the- Art. *IEEE Transactions on Evolutionary Computation*, 15(1), pp. 4-31.

[9] Dash, P. K., Liew, A. C., and Satpathy, H. P. (1999). A Functional Link Neural Network for Short Term Electric Load Forecasting. *Journal of Intelligent and Fuzzy Systems*, 7(3), pp. 209–221.

[10] Dash, P. K., Satpathy, H. P., Liew, A. C., and Rahman, S. (1997). Real-Time Short-Term Load Forecasting System Using Functional Link Network. *IEEE Transactions on Power Systems*, 12(2), pp. 675-68.

[11] Dash, S. K., Behera, A., Dehuri, S., and Cho, S.-B. (2013). Differential Evolution Based Optimization of Kernel Parameters in Radial Basis Function Networks for Classification. *International Journal of Applied Evolutionary Computation*, .4(1), pp. 56-80.

[12] Dehuri, S., Mishra, B. B., & Cho, S. B. (2008). Genetic feature selection for optimal functional link artificial neural network in classification. In *Intelligent Data Engineering and Automated Learning–IDEAL 2008*, pp. 156-163.

[13] Dehuri, S. and Cho, S.-B. (2009). A Comprehensive Survey on Functional Link Neural Networks and an adaptive PSO--BP learning for CFLNN. *Neural Computing and Applications*, 19(2), pp. 187-205.

[14] Dehuri, S. and Cho, S.-B. (2010). Evolutionary Optimized Features in Functional Link Neural Networks for Classification. *Expert Systems with Applications*, 37(6), pp. 4379-4391.

[15] Dehuri, S., Roy, R., Cho, S.-B., and Ghosh, A. (2012). An Improved Swarm Optimized Functional Link Artificial Neural Network (ISO-FLANN) for Classification. *The Journal of Systems and Software*, 85, pp.1333-1345.

[16] Derrac, J., Garcia, S., and Herrera, F. (2010). A Survey on Evolutionary Instance Selection and Generation. *International Journal of Applied Meta-heuristic Computing*, 1(1), pp. 60-92.

[17] Forerest, S. (1993). Genetic Algorithms: Principles of Natural Selection Applied to Computation. *Science*, 261, pp. 872-888.

[18] Frank, A. and Asuncion, A. (2010). *UCI Machine Learning Repository http://archive.ics.uci.edu/ml)*, Irvine, CA: University of California, School of Information and Computer Science.

[19] Goldberg, D. E. (1989). Genetic Algorithm in Search Optimization and Machine Learning, *Addison-Wesley*.

[20] Haring, B. and Kok, J. N. (1995). Finding Functional Links for Neural Networks by Evolutionary computation. In: van de Merckt, T., et al. (eds) BENELEARN1995, *Proceedings of the Fifth Belgian– Dutch Conference on Machine Learning, Brussels, Belgium*, pp. 71–78

[21] Haring, S., Kok, J. N., and van Wezel, M. C. (1997). Feature Selection for Neural Networks Through Functional Links Found by evolutionary Computation. *In Advances in Intelligent Data Analysis (IDA-97).LNCS* 1280, pp. 199–210.

[22] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall.

[23] He, X., Zhang, Q., Sun, N., & Dong, Y. (2009). Feature selection with discrete binary differential evolution. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, Vol. 4, pp. 327-330).

[24] Hu, Y.-C. (2008). Functional Link Nets with Genetic Algorithm Based Learning for Robust Non-Linear Interval Regression Analysis. *Neurocomputing*. 72(7). pp. 1808-1816.

[25] Hussain, A., Soraghan, J. J., & Durrani, T. S. (1997). A new adaptive functional-link neural-network-based DFE for overcoming co-channel interference. *Communications, IEEE Transactions on*, *45*(11), 1358-1362.

[26] Karegowda, A. G., Manjunath, A. S., and Jayaram, M. A. (2010). Comparative Study of Attribute Selection Using Gain Ratio and Correlation Based Feature Selection. *International Journal of Information Technology and Knowledge Management*, 2(2), pp. 271-277.

[27] Khusba Rami, N., Ahmed, A., and Adel, A. (2008). Feature Subset Selection Using Differential Evolution. In *Proceedings of 15th International conference on Advances in Neuro-Information Processing, Springer-Verlag, Berlin, Heidelberg, part-I*, pp.103-110,.

[28] Khushaba, R. Al-Ani, A., AlSukker, A., and Al-Jumaily, A. (2008). A Combined Ant Colony and Differential Evolution Feature Selection Algorithm. *Ant Colony Optimization and Swarm Intelligence*, pp. 1-12.

[29] Khushaba, R., Al-Ani, A., and Al-Jumaily, A. (2011). Feature Subset Selection using Differential Evolution and a Statistical Repair Mechanism. *Expert Systems with Applications*, 38(9), pp. 11511-11526.

[30] Klasser, M.S. and Pao, Y. H. (1988). Characteristics of the Functional Link Net: A Higher Order Delta Rule Net. *IEEE Proceedings of 2nd Annual International Conference on Neural Networks*, *San Diago, CA*, pp.507-513,

[31] Krishnaiah, D., Prasad, D. M. R., Bono, A., Pandiyan, P. M., and Sarbatly, R. (2008). Application of Ultrasonic Waves Coupled with Functional Link Neural Network for Estimation of carrageenan Concentration. *International Journal of Physical Sciences*, 3(4), pp. 90–96.

[32] Liu, H. and Motoda, H. (2002). On Issues of Instance Selection. *Data Mining and Knowledge Discovery*, 6, pp. 115-130.

[33] Majhi, B. and Shalabi, H. (2005). An Improved Scheme for Digital Watermarking using Functional Link Artificial Neural Network. *Journal of Computer Science*, 1(2), pp. 169–174.

[34] Marcu, T. and Koppen-Seliger, B. (2004). Dynamic Functional Link Neural Networks Genetically Evolved Applied to System Identification. In *Proceedings of ESANN'2004, Bruges (Belgium)*, pp. 115–120.

[35] Michalewicz, Z. (1998). *Genetic Algorithm + Data structure = Evolution Programs*. Springer Verlag, New York.

[36] Misra, B. B. and Dehuri, S. (2007). Functional Link Neural Network for Classification Task in Data mining. *Journal of Computer Science*, 3(12), pp. 948–955.

[37] Nayak, S. C., Misra, B. B., and Behera, H.S. (2012). Index prediction with Neuro-Genetic Hybrid network: A Comparative Analysis of Performance Computing. In *Proceedings of International Conference on Communication and Applications (ICCCA)*, pp. 1-6.

[38] Nayak, S. C., Misra, B. B., and Behera, H. S. (2012). Evaluation of Normalization Methods on Neuro-Genetic Models for Stock index Forecasting. In *Proceedings of 2012 World Congress on Information and Communication Technologies (WICT)*, pp. 602-607.

[39] Panagiotopoulos, D. A., Newcomb, R. W., and Singh, S. K. (1999). Planning with a Functional Neural Network Architecture. *IEEE Trans. Neural Network*, 10(1), pp. 115–127.

[40] Pao, Y.-H., Phillips, S. M. (1995). The Functional Link Net Learning Optimal Control. *Neurocomputing*, 9(2), pp. 149–164.

[41] Pao, Y. H. and Takefuji, Y. (1992). Functional Link Net Computing: Theory, system, Architecture and Functionalities. *IEEE Computer Journal*, pp. 76–79.

[42] Pao, Y. H., Phillips, S. M., and Sobajic, D. J. (1992). Neural-net Computing and Intelligent Control Systems. *International Journal of Control*, 56(2), pp. 263–289.

[43] Park, G. H. and Pao, Y. H. (2000). Unconstrained Word-Based Approach for Off-Line Script Recognition using Density Based Random Vector

Functional Link Net. *Neurocomputing*, 31(1), pp. 45–65.

[44] Patra, J. C., Pal, R. N., Baliarsingh, R., and Panda, G. (1999). Non-Linear Channel Equalization for QAM Signal Constellation using Artificial Neural Networks. *IEEE Transactions on Systems, Man, Cybernetics-Part B: Cybernetics*, 29(2), pp. 262–271.

[45] Patra, J. C. and van den Bos, A. (2000). Modelling of an Intelligent Pressure Sensor using Functional Link Artificial Neural Networks. *ISA Transaction*, 39(1), pp. 15–27.

[46] Patra, J. C., Goutam, C., and Subahas, M. (2008). Functional Link Neural Networks-Based Intelligent Sensors for Harsh Environments. *Sensors and Transducers Journal*, vol. 90, pp. 209–220.

[47] Patra, J. C. and Pal, R. N. (1995). A Functional Link Neural Network for Adaptive Channel Equalization. *Signal Processing*, 43(2), pp. 181–195.

[48] Patra, J. C., Pal, R. N. Chatterji, B. N., and Panda, G. (1999). Identification of Nonlinear Dynamic Systems using Functional Link Artificial Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(2), pp. 254-262.

[49] Patra, J. C. and Kot, A. C. (2002). Nonlinear Dynamic System Identification using Chebyshev Functional Link Artificial Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(4), pp. 505-511.

[50] Price, K., Storn, R., and Lampinen, J. (2005). Differential Evolution: *A Practical Approach to Global Optimization*, Springer-Verlag.

[51] Purwar, S., Kar, I. N., and Jha, A. N. (2007). On-line System Identification of Complex Systems using Chebyshev Neural Networks. *Applied Soft Computing*, 7(1), pp. 364–372.

[52] Roy, R., Dehuri, S., and Cho, S. B. (2011). A Novel Particle Swarm Optimization Algorithm for Multi-objective Combinatorial Optimization Problem. *International Journal of Applied Meta-heuristic Computing*, 2(4), pp. 41-57.

[53] Sierra, A., Macias, J. A., and Corbacho, F. (2001). Evolution of Functional Link Networks. *IEEE Transactions on Evolutionary Computation*, 5(1), pp. 54–65.

[54] Sing, S. N., Srivastava, K. N. (2002). Degree of Insecurity Estimation in a Power System using Functional Link Neural Network. *European Transactions on Electrical Power*, 12(5), pp. 353–359.

[55] Storn, R., & Price, K. (1995). *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces* (Vol. 3). Berkeley: ICSI.

[56] Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4), 341-359.

[57] Weng, W. D., Yong, C. S., and Lin, R. C. (2007). A Channel Equalizer using Reduced Decision Feedback Chebyshev Function Link Artificial Neural Networks. *Information Sciences*, 177(13), pp. 2642–2654.

[58] Weng, W. D., Yen, C. T. (2004). Reduced Decision Feed-back FLANN Non-Linear Channel Equaliser for Digital Communication Systems. *IEEE Proceedings-Communications*, 151(4), pp. 305–311.

[59] Yan, Z., Wang, Z. and Xie, H. (2008). The Application of Mutual Information Based Feature Selection and Fuzzy LS-SVM based Classifier in Motion Classification. *Computer Methods and Programs in Biomedicine*, vol.90, pp. 275-284.

[60] Zhang, G. P. (2000). Neural Networks for Classification: A Survey. *IEEE Transactions on Systems, Man, Cybernetics-Part C: Application and Reviews*, 30(4), pp. 451–461.

[61] Zhang, J. R., Zhang, J., Lok, T. M., and Lyu, M. R. (2007). A Hybrid Particle Swarm Optimization-Back Propagation Algorithm for Feed-Forward Neural Network Training. *Applied Mathematics and Computation*, vol. 185, pp. 1026-1037.

[62] Ghosh, A., Datta, A., and Ghosh, S. (2013). Self-Adaptive Differential Evolution for Feature Selection in Hyperspectral Image Data. *Applied Soft Computing*, 13(4), pp. 1969-1977.

[63] Hu, Y. C. and Tseng, F. M. (2007). Functional-Link Net with Fuzzy Integral for Bankruptcy Prediction. *Neurocomputing*, 70(16), pp. 2959– 2968.