International Journal of Computational Intelligence Systems, Vol. 8, No. 3 (2015) 539-552

Towards Crafting a Smooth and Accurate Functional Link Artificial Neural Networks Based on Differential Evolution and Feature Selection for Noisy Database

Ch. Sanjeev Kumar Dash
Department of Computer Science,
Silicon Institute of Technology, Silicon Hills,
Patia, Bhubaneswar-751024, Odisha, India
Email : sanjeev_dash@yahoo.com

Satchidananda Dehuri
Department of Systems Engineering,
Ajou University, San 5, Woncheon-dong,
Yeongtong-gu, Suwon-443-749, South Korea
Email: 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
Email: sbcho@yonsei.ac.kr

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

Received 9 March 2014

Accepted 8 December 2014

Abstract

This work presents an accurate and smooth functional link artificial neural network (FLANN) for classification of noisy database. The accuracy and smoothness of the network is taken birth by suitably tuning the parameters of FLANN using differential evolution and filter based feature selection. We use Qclean algorithm for identification of noise, information gain theory for filtering irrelevant features, and then supplied the remaining relevant attributes to the functional expansion unit of FLANN, which in turn map lower to higher dimensional feature space for constructing a smooth and accurate classifier. In specific, the differential evolution is used to fine tune the weight vector of this network and some trigonometric functions are used in functional expansion unit. The proposed approach is validated with a few benchmarking highly skewed and balanced dataset retrieved from University of California, Irvine (UCI) repository with a range of 5-20% noise. The insightful experimental study signifies the propensity of noise in the classification accuracy of a database with a range of noise from 5-20%. Moreover, our method suggests that noisy samples along with irrelevant set of attributes are deceptive and weakening the reliability of the classifier, therefore, it is required to reduce its effect before or during the process of classification.

Keywords: Differential evolution, Functional link artificial neural network, Classification, Feature selection

1 Introduction

The noise filtering or data filtering is conducted to eliminate the irrelevant content from the real data. This procedure is very significant but it is nearly impossible to get rid of the noise from all real data. Moreover, it is the first thing to consider when using the data for processing requiring accuracy and precision. The system needs to learn from the raw data itself by applying various state-oft-the-art techniques which is absolutely challenging. Although we have carried out our experimentation with the datasets borrowed from KEEL/UCI repository (not big data) for validation but it can be extended to big data.

Neural networks [42, 70], have emerged as an important tool for classification. Pao et al. [48], shows a direction that their proposed FLANN may be conveniently used for function approximation and can be extended for classification with faster convergence rate and lesser computational load than a multi-layer perceptron (MLP) structure. With this motivation, several classifiers have already been designed e.g., adaptive Particle Swarm Optimization-Back-propagation (PSO-BP) learning [19], Improved Swarm Optimized FLANN (ISO-FLANN) [21], etc. 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.

Furthermore, it is being accepted that the accuracy of the discovered model (i.e., a neural networks (NNs)[28], rules [13], decision tree [11], FLANN [21]), strongly depends on the quality of the data being mined. Hence, feature selection and noise removal are some of the preprocessing tasks to obtain quality data and hence brings lots of attention of many researchers [10, 69]. It is the process of selecting a subset of available features and removing noise to use in empirical modeling. Due to feature selection, we are not only increasing the accuracy but also increasing the facility to handle big data. Similarly, the removal of noise not only avoids the deception, it used to increase the reliability of the classifier as well. Gamberger et al. [3], have proposed a technique in which noisy examples are detected and eliminated from the training set, and a hypothesis is then built from the set of remaining examples. Verbaeten et al. [2], have used ensemble methods to identify noisy training examples. They considered the problem of mislabeled training examples in classification tasks, and address this problem by pre-processing the training set, i.e., by identifying and removing outliers from the training set. Xiong et al. [7], have presented a systematic evaluation on the effect of noise in machine learning. Instead of taking any unified theory of noise to evaluate the noise impacts, they differentiate noise into two categories: class noise and attribute noise, and analyze their impacts on the system performance separately.

Like feature selection, instance selection [39], is to choose a subset samples to achieve the original purpose of classification tasks, as if the whole dataset is used. Many variants of evolutionary and non-evolutionary based approaches are discussed in [22] for interested reader. The ideal outcome of instance selection is a model independent, minimum sample of data that can accomplish tasks with little or no performance deterioration. However, in this work, we restrict ourselves with feature selection only. 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) [9]. In this 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. As we know the architectural complexity of FLANN [18] is directly proportional to 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 [35, 36]) using gain ratio and then applying the usual procedure of function expansion and training by differential evolution.

This paper is set out as follows. Section 2 gives related works, overview of FLANN classifier, noise detection, 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.

2 Background

The background of this research work is presented in this section. In Subsections 2.1 and 2.2 review of literatures and FLANN classifier are discussed, respectively. Feature selection and its importance is 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 Klassen and Pao in [37]. Despite their linear nature, FLANNs can capture non-linear input–output relationships, provided that they 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 an n-dimensional representation space, which are constructed out of the original input, attributes [48]. FLANNs can be used for non-linear prediction and classification. In this connection, Subsections 2.1.1 and 2.1.2 are briefing some related works on FLANNs for classification and non-linear prediction.

2.1.1 FLANNs for Classification

In [63], a genetic algorithm for selecting an appropriate number of polynomials as a functional input to the network has been proposed by Sierra et al. and applied to the classification problem. However their main concern was selection of 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. Misra and Dehuri [43], 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 [31], has used the functional link net known as BpFLANN for classification of bankruptcy prediction. With a motivation to restrict certain limitations, Dehuri, et al. [18], have coupled genetic algorithm based feature selection with FLANN (GFLANN). In the sequel, Dehuri and Cho [19], have given a road map to FLANN and also designed a new PSO-BP adaptive learning mechanism for FLANN. In Dehuri and Cho [20], have contributed another stimulating work on FLANN [10], and then in succession an improved swarm optimized FLANN for classification has been proposed by Dehuri, et al. [21].

2.1.2 FLANNs for Prediction

Pao et al. [47] have presented a functional link neural network (CoFLANN) to learn the control systems. The reported results have several beneficial properties than generalized delta rule net with hidden layer and BP learning. Haring et al. [26] have proposed an algorithm (ClFLANN) that uses 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. [56] have proposed a CeFLANN and applied to the problem of channel equalization in a digital communication channel. It relies on BP-learning algorithm. Haring et al. [27] have proposed a ClaFLANN for different ways to select and transform features using evolutionary computation and shows that this kind of selection of features is a special case of so-called functional links. Hussain et al. [32] have described a new approach for the 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-errorate (BER) performance characteristics compared to the conventional methods. Chen et al. [12] 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. [15] have proposed an ElfFLANN with trigonometric basis functions to forecast the short-term electric load. Panagiotopoulos et al. [46] have reported better results by applying FLANN for planning in an interactive environment between two systems: the challenger and the responder. Patra et al. [51, 52] have proposed a FLANN with BP learning (SiFLANN) for identification of non-linear dynamic systems. Moreover, Patra et al. [52] have used FLANN to adaptive channel equalization in a digital communication system with 4-QAM signal constellation named as QsFLANN. They 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 [51, 52, 53], Patra et al. [53], 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 [50] have reported the performance of a holistic-styled word-based approach to off-line recognition of English language script. Authors 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) and it is helpful in improving the performance of the word recognition. A Chebyshev functional link artificial neural networks (CFLANN) has proposed by Patra et al. [54], for non-linear dynamic system identification. Sing et al. [64] have estimated the degree of insecurity in a power system by the proposed IeFLANN with a set of orthonormal trigonometric basis functions. In [41], an evolutionary search of genetic type and multi-objective optimization 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 equaliser in digital communication systems is proposed by Weng et al. [68]. Authors reported that the use of direct decision feedback can greatly improve the performance of FLANN structures. Majhi et al. [40], has applied FLANN for digital watermarking (DwFLANN), their results shows that FLANN has better performance than other algorithms in this line. Purwar et al. [61] have proposed a Chebyshev functional link neural network (SyiFLANN) for system identification of unknown dynamic non-linear discrete time systems. Weng et al. [67], has proposed a reduced decision feed-back Chebyshev functional link artificial neural networks (RDF-CFLANN) for channel equalization. Two simple modified CcFLANNs are proposed by Krishnaiah et al. [38], for estimation of carrageenan concentration. In the first model, a hidden layer is introduced and trained by EBP. In the second model, functional links are introduced to the neurons in the hidden layer and it is trained by EBP. In [55], a FLANN with trigonometric polynomial functions (IsFLNN) are used in intelligent sensors for harsh environment that effectively linearizes the response characteristics, compensates for non-idealities 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 [30], 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.

2.2 FLANN Classifier

The FLANN architecture [52, 54, 56, 16], 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 FLNN 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 then to find the weight parameters W that provides the best possible approximation of f on the set of input-output examples. Hence 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) $\Phi_1 = 1$, (ii) the subset $\gamma_j = \{\Phi_i \in \gamma\}_{i=1}^{j}$ is 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:

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

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., $\hat{y} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_m]^T$ 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-dimensional output vector, equation (1) can be written in matrix notation as follows.

$$S = W\Phi, \qquad (2)$$

where W is a $(m \times N)$ 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

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

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 jth output of the FLANN at iteration k is given by

$$\hat{y}(k) = \rho(\sum_{i=}^{N} (w_{ji}(k)\Phi_i(X_k)) = \rho(w_j(k)\Phi^T(X_k)) \qquad (4)$$

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 to the set of functions considered for expansion of the input dataset.

However, dimensionality of many problems itself are 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.

2.3 Noise Detection

The two types of noise can be distinguished [5,6]: i) Noise in attributes: It is given by the errors occurred during the entrance of the values of the attributes. Among the sources of this type of noise are variable with missing values, and redundant data; ii) Noise in classes: It is given by the errors introduced during the assignment of the instances to the classes. The presence of this kind of noise may be due to subjectivity, errors in the data entry process, and incorrect information for assigning an instance to a class.

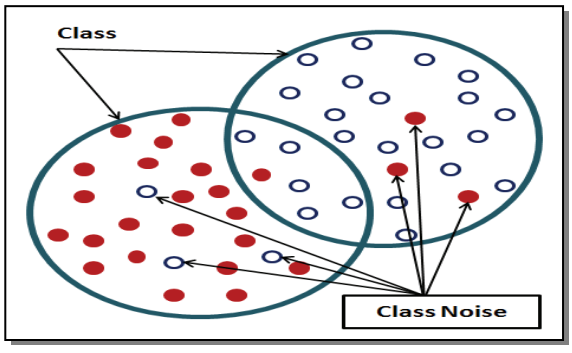

Figure 1. Graphical Representation of Class Noise

Let us discuss the Qclean algorithm for removal of noise in class level. According to the definition of class noise [5], the noisy instances will be those that are badly located in the cloud of points that define each of the classes (see Figure 1).

We introduce a measure to evaluate the quality of an instance. Thus, for the ith instance, we compute $Q_i = (r_i - d_i) / \max(d_i, r_i)$, where d_i is the distance of the ith instance to the centroid of its class and r_i is the minimum distance of the ith instance to the centroid of the classes where it does not belong to. A noisy instance will have negative values for the quality measure Q. However, some instances located near to the boundary of two or more classes may also have small negative values for the quality measure.

First, the quality measure Q is computed for each instance of the training set. Then, each instance with Q<0 is regarded as a candidate to be a noisy instance. The next step is to find out the k nearest neighbors to each noisy instance candidate. Finally, we count the number of neighbors that belong to the same class. If the majority of the neighbors does not belong to the same class the candidate instance is then regarded as noisy and discarded from the training set. The algorithm is given below:

```
QcleanNOISE(Training Dataset D)
{
    For( each instance I_i in D)
        Find out its quality measure Q( I_i )
    endFor
        CandNoise = Φ
        For( each instance I_i in E)
            if ( Q( I_i ) < 0 )
        CandNoise = CandNoise U { I_i }
        endFor
        For( each instance I_i in CandNOISE)
Find out its k nearest neighbors NN in D
    Count( I_i ) =0
For( each NN of I_i )
    if (Class(NN) ==Class( I_i ) )
    Count = Count +1
    endFor
    For( each instance I_i in CandNOISE)
        if (Count( I_i ) < (k+1)/2 ) D = D - { I_i }
        endFor
Return E
}
```

2.4 Feature Selection

Feature selection (FS) [34, 35, 36], is essentially a task to remove irrelevant and/or redundant features. In other words, feature selection techniques study how 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: It precedes the actual classification process. 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 feature selected depends on the classifier model used [33]. 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) [60, 65, 66],is a population based meta-heuristic search algorithm which typically operates on real valued chromosome encodings. Like GAs, DE maintains a pool of potential solutions which are then perturbed in an effort to uncover 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 scaled difference of two randomly chosen individuals of the current population. One of the advantages of this approach is that the resulting 'step' size and orientation during the perturbation process automatically adapts to the fitness function landscape.

Over the years, there are many variants of DE algorithms developed [60, 14], however, we primarily describe a version of the algorithm based on the DE/rand/1/bin scheme [65]. The different variants of the DE algorithm are described using the shorthand DE/x/y/z, where x specifies how the base vector (of real values) is chosen (rand if it is randomly selected, or best if the best individual in the population 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 population of n, d-dimensional vectors $x_i = (x_{i1}, x_{i2}, .., x_{id}),\ i = 1...n$ each of which encode a solution is randomly initialized and evaluated using a fitness function f(.). During the search process, each individual (i) is iteratively refined. The following three steps are required while execution.

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

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.

Under the mutation operator, for each vector $x_i(t)$ a donor vector $v_i(t+1)$ is obtained by equation (5).

$$v_i(t+1) = x_k(t) + f_m * (x_l(t) - x_m(t)), \qquad (5)$$

where k, l, m ∈ 1…n are mutually distinct, randomly selected indices, and all the indices ≠ i $(x_k(t))$ is referred to as the base vector and $(x_l(t) - x_m(t))$ is referred as difference vector). Selecting three indices randomly implies that all members of the current population have the same chance of being selected, and therefore influencing the creation of the difference vector. The difference between vectors x_l and x_m is multiplied by a scaling parameter f_m called mutation factor and a range for the parameter must be associated to it, that is $f_m \in [0, 2]$. The mutation factor controls the amplification of the difference between x_l and x_m which is 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 [60, 14]).

A notable feature of the mutation step in DE is that it is self-scaling. The size/rate of mutation along each dimension stems solely from the location of the individuals in the current population. The mutation step self-adapts as the population converges leading to a finer-grained search. In contrast, the mutation process in GA is typically based on (or draws from) a fixed probability density function.

Following the creation of the donor vector, a trial vector $u_i(t+1) = (u_{i1}, u_{i2}, u_{i3}, ..., u_{id})$ is obtained by equation (6).

$$u_i(t+1) = \begin{cases} v_{ip}(t+1), & if\ (rand \le c_r)\ or\ (i = rand(ind)) \\ x_{ip}(t), & if\ (rand > c_r)\ and\ (i \ne rand(ind)) \end{cases}, \qquad (6)$$

where $p = 1,2,\ldots,d,$ 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(ind) is a randomly chosen index, selected from the range (1, 2, … ,d). The random index is used to ensure that the trial vector differs by at least one element from $x_i(t)$. 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.

Finally, if the fitness of the trial vector exceeds that of the fitness of the parent then it replaces the parent as described in equation (7).

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

Price and Storn [60], provide 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.

3 Proposed Method

With an objective to design a smooth and accurate classifier, the proposed approach is combining the idea of combating noise, feature selection, and simple FLANN classifier [21]. It is a three phase method. In phase one, we are removing noise by Qclean [4], algorithm. In second phase, we are selecting a set of relevant features by using the entropy measure and in the third phase the weights of FLANN are trained using differential evolution. Figure 2 depicts the overall architecture of the approach.

In first phase one we are removing noise from class level using Qclean algorithm [4]. In the second phase, we rank the features or attributes according to information gain ratio and then delete an appropriate number of features which have the least gain ratio[9]. 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,

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

where p_i is the non-zero probability that an arbitrary tuple in D belongs to class C_i and is estimated by $\left|C_{i,D}\right| / \left|D\right|$. 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.

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

where v is the number of distinct values in the attribute A, $\left|D\right|$ is the total number of tuples in D and $\left|D_j\right|$ 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)

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

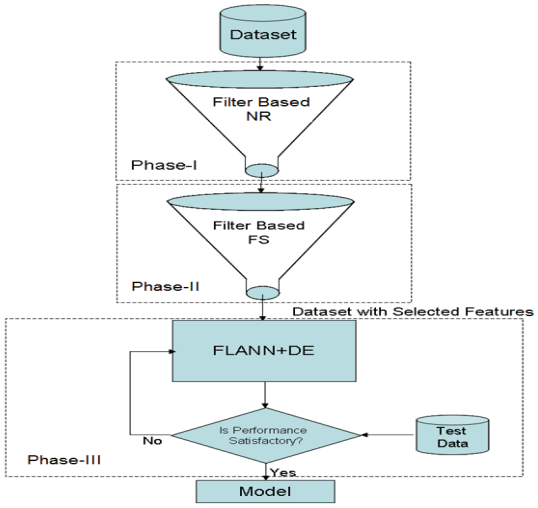

Figure 2: Architecture of Proposed Method

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

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

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.

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

The more the gain ratio of an attribute the more important it is. In our approach, first, the feature selection is done using information gain ratio and then the reduced dataset is used for automatic training and determination of the parameters of FLANN using DE.

In the third phase, we are focusing on the learning of the classifier. As mentioned there are two steps within the learned procedure. In step one 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.

Suppose the maximum number of trigonometric function is set to $K_{\max}$, then the structure of the individual is represented as follows (Figure 3):

Figure 3: Structure of the Individual

In other words each individual has two constituent parts such as weight and bias. The length of the individual is $2K_{\max}+1$.

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

$$f(x) = \frac{1}{N}\sum_{1}^{N}\left(t_n - \Phi(\vec{x_i})\right)^2 \quad (13)$$

where N is the total number of training sample t_n is the actual output and is the estimated output of FLNN.. Figure 4 illustrates the two step learning procedure adopted in this work.

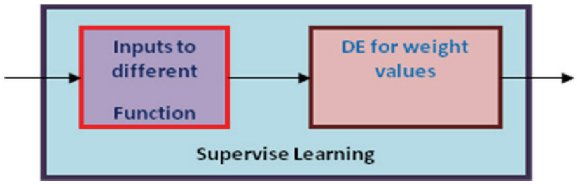

Figure 4: Two step learning scheme

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 centers, width, and bias are created.

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

where $d = 2K_{\max}+1$ and t is the iteration number..

At the start of the algorithm this n_p set of individuals are 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):

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

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,…, n_p }, $r_1 \neq r_2 \neq r_3 \neq i$.

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

$$u_i^{(t+1)} = u_{i1}^{(t+1)}, u_{i2}^{(t+1)}, ..., u_{id}^{(t+1)}$$

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

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 on 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):

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

4 Experimental Study

The dataset and experimental parameter setting is discussed in subsection 4.1. Results are analyzed in subsection 4.2.

4.1 Description of Dataset and Parameters

The dataset used to test the proposed method were obtained from the UCI machine learning repository [24]. Four balanced and unbalanced datasets have been chosen to validate the proposed method. The details about the four datasets are given below.

Iris: This dataset 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 belongs to class 3. None of the attributes contain any missing values. All attributes are continuous.

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 in to different classes are almost balanced.

Pima: This dataset 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.

Ionosphere: The dataset is referred from UCI repository for Machine Learning database. There are 351 instances, 33 attribute and 2 classes. Class 1 has 225 instances and Class 2 has 126 instance (Figure 10). None of the attributes contain any missing values.

In our experiment, every dataset is randomly divided into two mutually exclusive parts: 60% as training sets and 40% as test sets. The parameters' value used for validating our proposed method is listed in Table 2. These parameters were obtained after several rounds of independent runs. However, the number of iterations are varies from dataset to dataset.

Table 2: Parameters used for simulation

Parameter	Iris	Statlog (Heart)	Pima	Ionosphere
Population	50	50	50	50
Mutation	0.2	0.3	0.4	0.4
Crossover	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 Statlog (Heart) and Pima. In the case of Ionosphere the ideal number of iteration can varies from 100~200.

4.2 Result Analysis

The experimental test results are presented in Tables 3-5. The five cross validation is used to show the classification accuracy of 5%, 10%, 15%, and 20% from a noisy dataset.

Table 3: Results Obtained From DE-FLANN for Iris Dataset

Number of Folds	DE-FLANN (5% noise)	DE-FLANN (10% noise)	DE-FLANN (15% noise)	DE-FLANN (20% noise)
1st	96.43	96.43	89.28	85.71
2nd	96.67	96.67	96.67	96.67
3rd	90.00	88.89	93.33	93.33
4th	93.33	90.00	90.00	90.00
5th	96.67	93.33	93.33	93.33
Avg.	94.62	93.06	92.53	91.81

Table 4: Results Obtained From DE-FLANN for Heart Dataset

Number of	DE-FLANN	DE-FLANN	DE-FLANN	DE-FLANN

Folds	(5% noise)	(10% noise)	(15% noise)	(20% noise)
1st	83.33	83.33	85.18	81.48
2nd	88.89	87.04	88.89	87.04
3rd	79.63	77.78	77.78	77.78
4th	74.07	74.07	70.37	72.22
5th	92.59	92.59	83.33	85.18
Avg.	83.70	82.96	81.11	80.74

Table 5: Results Obtained From DE-FLANN for Pima Dataset

Number of Folds	DE-FLANN (5% noise)	DE-FLANN (10% noise)	DE-FLANN (15% noise)	DE-FLANN (20% noise)
1st	75.32	74.67	73.38	72.73
2nd	72.73	74.03	73.38	71.43
3rd	74.67	68.18	68.18	69.48
4th	75.82	73.20	73.20	73.86
5th	74.51	72.55	72.55	67.32
Avg.	74.61	72.53	71.74	70.96

Table 6: Results Obtained From DE-FLANN for Ionosphere Dataset

Number of Folds	DE-FLANN (5% noise)	DE-FLANN (10% noise)	DE-FLANN (15% noise)	DE-FLANN (20% noise)
1st	85.91	81.69	84.51	81.69
2nd	84.28	81.43	82.86	81.43
3rd	85.71	84.28	75.71	72.86
4th	90.00	94.28	91.43	88.57
5th	78.57	77.14	78.57	78.57
Avg.	84.90	83.77	82.61	80.62

Table 7: Filtered Attributes of Datasets

Dataset	Attributes of the Dataset	Attributes Removed
Iris	1~4	2
Statlog(heart)	1~13	1,4,6,7
Pima	1~8	1,3
Ionosphere	1~33	2,3,11,12,13,15, 17,19,20,22,30

Table 8: Results obtained from without Noise without feature selection and with Feature Selection

Dataset	Accuracy in Percentage after noise removal	Accuracy in percentage after noise
	without feature selection	removal and feature selection
Iris	98.33	98.33
Stalog(heart)	86.57	86.57
Pima	79.20	79.20
Ionosphere	86.7213	97.3333

The error rate varies over number of iterations for Iris, Pima, Statlog (Heart), and Ionosphere is illustrated in Figures 5, 6, 7, and 8, respectively.

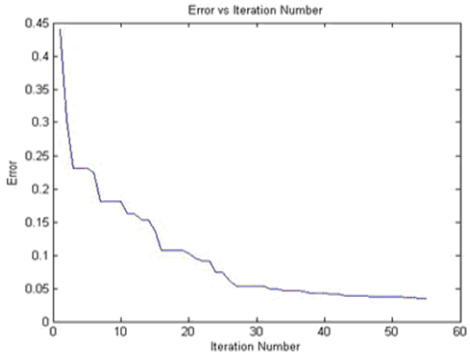

Figure 5: IRIS-Iteration vs. Error

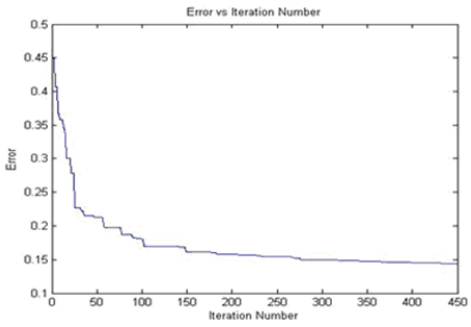

Figure 6: PIMA-Iteration vs. Error

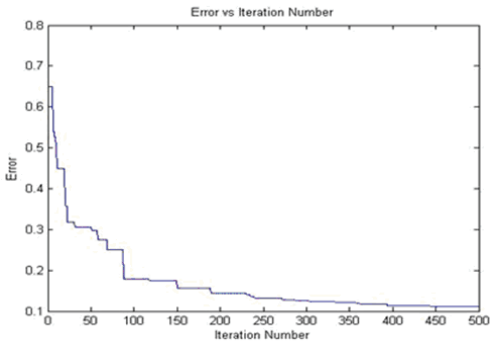

Figure 7: Statlog (Heart)-Iteration vs. Error

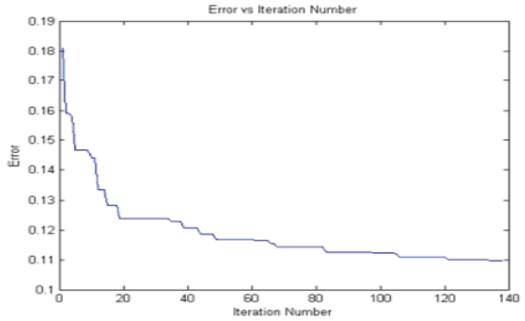

Figure 8: Ionosphere-Iteration vs. Error

Figure 9 illustrate the classification accuracy trends with respect to different noise level in the datasets.

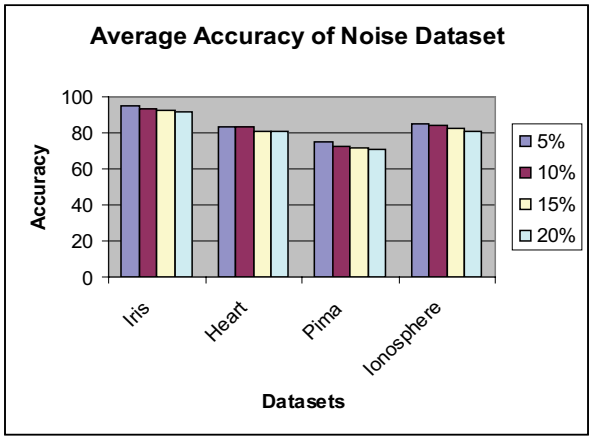

Figure 9: Classification Accuracy of DE-FLANN w.r.t Different Noisy Datasets

5 Conclusion

In this article, the integrated framework of differential evolution and FLANN along with a filter based feature selection has been crafted to classify unknown patterns from noisy database. A Qclean based noise detection algorithm along with the above integrated framework is functioning towards combating 5%, 10%, 15%, and 20 % of noise in a database. The experimental results confirm that this combination of filter based feature selection and training of FLANN using differential evolution obtains accurate and smooth classification accuracy. Further, we have observed that if we reduce 1/3rd of total attributes by the process of filter approach, then the network shows an improvement and or constancy in accuracy. The effect of noise has been realized from the experimental set up and observed that it is required to combat with the noise before or in the process of classification for developing a reliable classifier with acceptable performance.

References

1. H. Xiong G. Pandey, M. Steinbach and V. Kumar, Enhancing Data Analysis with Noise Removal, *IEEE Transactions on Knowledge and Data Engineering*,(2006),18(3),304-319.
2. S. Verbaeten and A. Van Assche , Ensemble Methods for Noise Elimination in Classification Problems, *Multiple Classifier Systems*,(2003), 317-325.
3. D. Gamberger, N. Lavrac, and S. Dzeroski, Noise Detection and Elimination in Data Preprocessing: Experiments in Medical Domains, *Applied Artificial Intelligence*,(2000), 14(2), 205-223.
4. L. Daza, and E. Acuna, An Algorithm for Detecting Noise on Supervised Classification, *in Proceedings of WCECS-07, the 1st World Conference on Engineering and Computer Science*,(2007), pp. 701-706.
5. X. Zhu, X. Wu and Q. Chen, Eliminating Class Noise in Large Datasets, *in ICML* ,(2003), vol. 3, pp. 920-927.
6. X. Zhu, X. Wu , and Q. Chen , Bridging Local and Global Data Cleansing: Identifying Class Noise in Large, Distributed Data Datasets, *Data mining and Knowledge Discovery*,(2006), 12(2-3), 275-308.
7. H. Xiong, G. Pandey, M. Steinbach and V. Kumar, Enhancing Data Analysis with Noise Removal". *IEEE Transactions on Knowledge and Data Engineering*, (2006), 18(3), 304-319.
8. I-A. Abu-Mahfouz, A Comparative Study of Three Artificial Neural Networks for the Detection and Classification of Gear Faults, *International Journal of General Systems*,(2005), 34(3), 261–277.
9. S.Aruna, L. V. Nandakishore and S.P. Rajagopalan,. (2012), Hybrid Feature Selection Method based on IGSBFS and Naïve Bayes for the Diagnosis of Erythemato - Squamous A Diseases, *International Journal of Computer Applications,*(2012), 41(7).
10. R. Battiti, Using Mutual Information for Selecting Features in Supervised Neural Network Learning, *IEEE Transactions Neural Networks*, (1994), 5(4), 537-549.
11. D.R. Carvalho and A. A. Freitas , A Hybrid Decision Tree/Genetic Algorithm Method for Data Mining, *Information Sciences*,(2004), 163(1-3),13-35.
12. CLP. Chen et al., An Incremental Adaptive Implementation of Functional Link Processing for Function Approximation, Time Series Prediction, and System Identification, *Neurocomputing*, (1998), 18(1), 11–31.
13. M. Das, R. Roy, S. Dehuri, and S.-B .Cho. , A New Approach to Associative Classification Based on Binary Multi-objective Particle Swarm Optimization, *International Journal of Applied Meta-heuristic Computing*, (2011), 2(2), 51-73 .

14. S. Das and P. N. Suganthan, Differential Evolution: A Survey of the State-of- the- Art, *IEEE Transactions on Evolutionary Computation*,(2011), 15(1), 4-31.
15. P. K. Dash, A. C. Liew and H. P. Satpathy, A Functional Link Neural Network for Short Term Electric Load Forecasting, *Journal of Intelligent and Fuzzy Systems*, (1999),7(3),209–221.
16. P.K Dash, H. P. Satpathy A.C. Liew, and S. Rahman, A Real-Time Short-Term Load Forecasting System using Functional Link Network, *IEEE Transactions on Power Systems*, ,(1997),12(2), 675—680.
17. C. S. K. Dash, A. Behera, S. Dehuri , and S.-B. Cho Differential Evolution Based Optimization of Kernel Parameters in Radial Basis Function Networks for Classification, *International Journal of Applied Evolutionary Computation*, (2013) ,4(1), 56-80.
18. S. Dehuri , B. B Mishra, and S.-B. Cho, Genetic Feature Selection for Optimal Functional Link Neural Network in Classification. *in: Fyfe C et al (eds) IDEAL, LNCS 5326*,(2008), pp.156–163.
19. S. Dehuri, and S.-B. Cho, A Comprehensive Survey on Functional Link Neural Networks and an adaptive PSO--BP learning for CFLNN, *Neural Computing and Applications*, (2009) , 19(2),187-205.
20. S. Dehuri, and S.-B. Cho, Evolutionary Optimized Features in Functional Link Neural Networks for Classification, *Expert Systems with Applications*, (2010) , 37(6), 4379-4391 .
21. S. Dehuri, R. Roy, S.-B. Cho and A. Ghosh, An Improved Swarm Optimized Functional Link Artificial Neural Network (ISO-FLANN) for Classification, *The Journal of Systems and Software*, (2012), 85(6), 1333-1345.
22. J. Derrac, S. Garcia, and F. Herrera, A Survey on Evolutionary Instance Selection and Generation, International Journal of Applied Meta-heuristic Computing, (2010). 1(1), 60-92.
23. S. Forerest, Genetic Algorithms: Principles of Natural Selection Applied to Computation, *Science*, (1993), 261, 872-888.
24. A. Frank, and A. Asuncion, UCI Machine Learning Repository (http://archive.ics.uci.edu/ml), Irvine, *CA: University of California*, *School of Information and Computer Science*, (2010).
25. D. E. Goldberg, Genetic Algorithm in Search Optimization and Machine Learning, *Addison-Wesley: Reading: MA*, (1989).
26. B. Haring, JN. Kok , 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*, (1995), pp.71–78
27. S. Haring, N. K. Joost, and C. V. W. Michiel, Feature Selection for Neural Networks Through Functional Links Found by evolutionary Computation, *Advances in Intelligent Data Analysis Reasoning about Data* .LNCS 1280, (1997), pp.199–210.
28. S. Haykin, Neural Networks: A Comprehensive Foundation, Upper Saddle River, *NJ: Prentice Hall*,(1994).
29. X. He, Q. Zhang, N. Sun, and Y. Dong, Feature Selection with Discrete Binary Differential Evolution. *International Conference on Artificial Intelligence and Computational Intelligence*, AICI'09, (2009), vol.4, pp.327-330.
30. Y-C. Hu, Functional Link Nets with Genetic Algorithm Based Learning for Robust Non-Linear Interval Regression Analysis, *Neurocomputing*. (2008),doi:10.1016/J.neucom.2008.07.002.
31. Y-C. Hu and F-M. Tseng, Functional-Link Net with Fuzzy Integral for Bankruptcy Prediction, *Neurocomputing*, (2007) , 70(16),2959– 2968.
32. A. Hussain, J. J. Soraghan, and T. S. Durrani, A New Adaptive Functional Link Neural Network Based DFE for Overcoming Co-Channel Interference, *IEEE Tranactions on Communication*, (1997), 45(11), 1358–1362.
33. A. G. Karegowda, A. S. Manjunath, and M. A. Jayaram, Comparative Study of Attribute Selection Using Gain Ratio and Correlation Based Feature Selection *International Journal of Information Technology and Knowledge Management*, (2010), 2(2),271-277.
34. R. N. Khusba, A. Ahmed, A. Adel, and et al. ,Feature Subset Selection Using Differential Evolution*, in Proceedings of 15th International conference on Advances in Neuro-Information Processing*, Springer-Verlag, Berlin, Heidelberg, part-I, (2008), pp.103-110,.
35. R. Khushaba, A. Al-Ani, A. AlSukker, and A. Al-Jumaily, A Combined Ant Colony and Differential Evolution Feature Selection Algorithm, *Ant Colony Optimization and Swarm Intelligence*, (2008) , pp.1-12 .
36. R. Khushaba, A. Al-Ani, and A. Al-Jumaily, Feature Subset Selection using Differential Evolution and a Statistical Repair Mechanism , *Expert Systems with Applications*, (2011) , 38(9), 11511-11526.
37. MS. Klasser, YH. Pao 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, (1988) ,pp.507-513,
38. D. Krishnaiah, D. R. Prasad, A. Bono, P. M. Pandiyan., and R. Sarbatly, Application of Ultrasonic Waves Coupled with Functional Link Neural Network for Estimation of carrageenan Concentration, *International Journal of Physical Sciences*, (2008) , 3(4),90–96.
39. H. Liu, and H. Motoda, On Issues of Instance Selection, *Data Mining and Knowledge Discovery*, (2002),vol.6, pp.115-130.
40. B. Majhi, and S. Hasan, An Improved Scheme for Digital Watermarking using Functional Link Artificial Neural Network, *Journal of Computer Science*, (2005) ,1(2), 169–174.
41. T. Marcu , B. Koppen-Seliger , Dynamic Functional Link Neural Networks Genetically Evolved Applied to System

Identification. *in: Proceedings of ESANN'2004*, Bruges (Belgium), (2004), pp. 115–120.
42. Z. Michalewicz, Genetic Algorithm + Data structure = Evolution Programs, *Springer Verlag*, New York, (1998).
43. B. B. Misra, S. Dehuri ,Functional Link Neural Network for Classification Task in Data mining, *J Comput Sci*, (2007) ,3(12), 948–955.
44. S. C. Nayak, B. B. Misra, and H. S. Behera, Index prediction with Neuro-Genetic Hybrid network: A Comparative Analysis of Performance Computing, *International Conference on Communication and Applications (ICCCA)*, (2012), pp.1-6.
45. S. C. Nayak, B. B. Misra, and H. S. Behera, Evaluation of Normalization Methods on Neuro-Genetic Models for Stock index Forecasting, *Information and Communication Technologies (WICT)*, 2012 World Congress, (2012), pp.602-607.
46. D. A. Panagiotopoulos, R. W. Newcomb, S. K. Singh, Planning with a Functional Neural Network Architecture". *IEEE Trans Neural Network*, (1999), 10(1),115–127.
47. Y-H. Pao, SM. Philips The Functional Link Net Learning Optimal Control, *Neurocomputing*, (1995), 9(2), 149–164.
48. Y-H. Pao, Y. Takefuji, Functional Link Net Computing :Theory, system, Architecture and Functionalities, *IEEE Comput*, (1992), pp.76–79.
49. Y.–H. Pao, et al., Neural-net Computing and Intelligent Control Systems, *International Journal of Control*, (1992), 56(2), pp.263–289.
50. GH. Park and YH. Pao Unconstrained Word-Based Approach for Off-Line Script Recognition using Density Based Random Vector Functional Link net, Neurocomputing, (2000) ,31(1), 45–65.
51. JC. Patra et al., Identification of Non-Linear Dynamic Systems using Functional Link Artificial Neural Networks, *IEEE Trans Syst Man Cyber Part B Cybern* , (1999),29(2),254–262.
52. JC. Patra et al., Non-Linear Channel Equalization for QAM Signal Constellation using Artificial Neural Networks, *IEEE Transactions on Systems, Man, Cybernetics-Part B: Cybernetics*, (1999),29(2), 262–271.
53. J. C. Patra, A. Van den Bos, Modeling of an Intelligent Pressure Sensor using Functional Link Artificial Neural Networks, *ISA Trans* , (2000),39(1) ,15–27.
54. J. C. Patra, and A. C Kot, Non-Linear Dynamic System Identification using Chebyshev Functional Link Artificial Neural Networks, *IEEE Trans on Systems*, Man, and Cybernetics, Part B: Cybernetics, (2002), 32(4),505–511.
55. JC. Patra et al., Functional Link Neural Networks-Based Intelligent Sensors for Harsh Environments, *Functional Link Neural Networks-Based Intelligent Sensors for Harsh Environments*, (2008),vol.90 ,pp.209–220.
56. J. C. Patra, N. R. Pal, A Functional Link Neural Network for Adaptive Channel Equalization, *Signal Process*, (1995) ,43(2),181–195.
57. J. C. Patra, R. N. Pal, B. N. Chatterji, and G. Panda, Identification of Nonlinear Dynamic Systems using Functional Link Artificial Neural Networks , *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*,(1999),29(2), 254-262.
58. J.C. Patra, and A.C. Kot, Nonlinear Dynamic System Identification using Chebyshev Functional Link Artificial Neural Networks ,*IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, (2002), 32(4), 505-511.
59. J. C. Patra and R. N. Pal, A Functional Link Artificial Neural Network for Adaptive Channel Equalization, *Signal Processing*, ,(1995), 43(2), 181-195.
60. K. Price, R. Storn and J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, *Springer-Verlag*, (2005).
61. S .Purwar et al., On-line System Identification of Complex Systems using Chebyshev Neural Networks, *Applied Soft Computing*, (2007),7(1),364–372.
62. R. Roy, S. Dehuri, , and S.-B. Cho, A Novel Particle Swarm Optimization Algorithm for Multi-objective Combinatorial Optimization Problem, *International Journal of Applied Meta-heuristic Computing*, (2011), 2 (4), 41-57.
63. A. Sierra, J. A. Macias, F. Corbacho, Evolution of Functional Link Networks, *IEEE Transactions on Evolutionary Computation*, (2001), 5(1),54–65.
64. S. N. Sing, K. N. Srivastava, Degree of Insecurity Estimation in a Power System using Functional Link Neural Network, *European Transactions on Electrical Power*, (2002), 12(5), 353–359.
65. R. Stron, and K. Price, Differential Evolution-A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-05-012: *International Computer Science Institute*, Berkely, (1995).
66. R. Stron, and K. Price, Differential Evolution- Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization*, (1997), 11(4),341-359.
67. W-D. Weng et al., A Channel Equalizer using Reduced Decision Feedback Chebyshev Function Link Artificial Neural Networks. Information Sciences, (2007),177(13),2642–2654.
68. W-D. Weng, CT. Yen. Reduced Decision Feed-back FLANN Non-Linear Channel Equaliser for Digital Communication Systems, *IEE Proceedings-Communications*, (2004),151(4),305–311.
69. Z. Yan, Z. Wang, and H. Xie, The Application of Mutual Information Based Feature Selection and Fuzzy LS-SVM based Classifier in Motion Classification, *Computer Methods and Programs in Biomedicine*, (2008), vol.90(3), 275-284.
70. G. P. Zhang, Neural Networks for Classification: A Survey". *IEEE Transactions on Systems, Man, Cybernetics-Part C: Application and Reviews*, (2000), 30(4), 451–461.