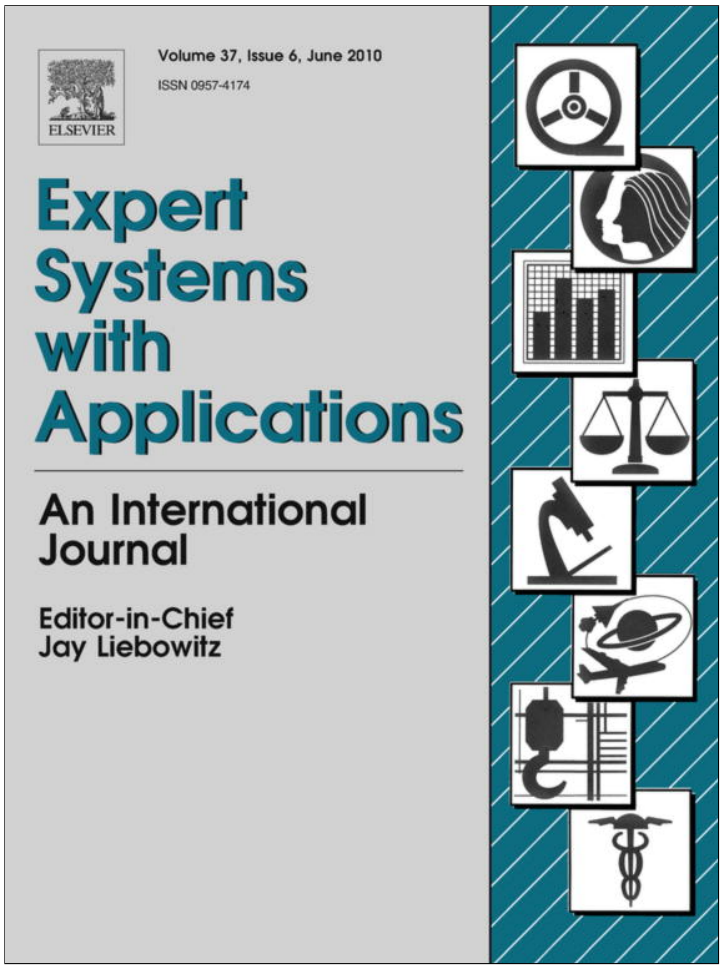

Expert Systems with Applications 37 (2010) 4379–4391

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Evolutionarily optimized features in functional link neural network for classification

Satchidananda Dehuri *, Sung-Bae Cho

Soft Computing Laboratory, Department of Computer Science, Yonsei University, 262 Seongsanno, Seodaemun-gu, Seoul 120-749, Republic of Korea

ARTICLE INFO

Keywords:
Classification
Data mining
Functional link neural network
Genetic algorithms
Radial basis function network

ABSTRACT

In this paper, an adequate set of input features is selected for functional expansion genetically for the purpose of solving the problem of classification in data mining using functional link neural network. The proposed method named as HFLNN aims to choose an optimal subset of input features by eliminating features with little or no predictive information and designs a more compact classifier. With an adequate set of basis functions, HFLNN overcomes the non-linearity of problems, which is a common phenomenon in single layer neural networks. The properties like simplicity of the architecture (i.e., no hidden layer) and the low computational complexity of the network (i.e., less number of weights to be learned) encourage us to use it in classification task of data mining. We present a mathematical analysis of the stability and convergence of the proposed method. Further the issue of statistical tests for comparison of algorithms on multiple datasets, which is even more essential in data mining studies, has been all but ignored. In this paper, we recommend a set of simple, yet safe, robust and non-parametric tests for statistical comparisons of the HFLNN with functional link neural network (FLNN) and radial basis function network (RBFN) classifiers over multiple datasets by an extensive set of simulation studies.

1. Introduction

For the past two decades, there have been a lot of studies focused on the classification problem in the field of data mining (Fayyad, Piatetsky-Shapiro, & Smyth, 1996; Kriegel, 2007). The abstract goal of data mining is to discover novel and useful information in databases this is where consensus ends and the means of achieving this goal are as diverse as the communities contributing. Classification is one of the fundamental tasks of data mining. Neural networks (Zhang, 2000, 2007) have emerged as an important tool for classification. The recent vast research activities in neural classification have established that neural networks are a promising alternative to various conventional classification methods. The ANN's are capable of generating complex mapping between the input and the output space and thus these networks can form arbitrarily complex nonlinear decision boundaries. This does not mean that finding such type of networks is easy. On the contrary, problems such as local minima trapping, saturation, weight interference, initial weight dependence, and overfitting make neural network training difficult. Moreover, most neural learning methods, being based on gradient descent, cannot search the non-differentiable landscape of multi-layer architectures. This is the key; since it can be proved that if a network is allowed to adapt its architecture, it can solve any learnable problem in polynomial time.

Myriad combinations of evolutionary algorithms (EAs) and neural networks (NNs) used in classification problem have been proposed (Branke, 1995; Cant-Paz & Kamath, 2005; Yao, 1999). Genetic algorithms have been used to train the networks (Caudell & Dolan, 1989; Fogel, Fogel, & Porto, 1990; Kitano, 1990; Montana & Davis, 1989; Whitley & Hanson, 1989), design their architecture (Miller, Todd, & Hegde, 1989), and select feature subsets (Oh, Lee, & Moon, 2004; Seidlecki & Skalansky, 1989; Yang & Honavar, 1998). However, most of these combinations have been devoted to neural networks with hidden layer but in this paper our primary focus is on combining genetic algorithms with neural networks having no hidden layer. FLNNs (Pao & Philips, 1995; Pao & Takefuji, 1992) a class of higher order neural networks are a network of such type. FLNNs can capture non-linear input–output relationships, provided that they are fed with an adequate set of functional inputs. The primary interest of considering such type of neural networks is to reduce architectural complexity and to achieve faster learning by optimizing less number of weights parameters. Pao (1992) have given a pointer that FLNN may be conveniently used for function approximation and can be extended for pattern classification with faster convergence rate and lesser computational load than a multi-layer perceptron (MLP) structure.

Furthermore, selecting optimal set of features, as the input for functional expansion is another interesting and improvement in this direction. Selecting optimal set of features and fed as the input for the functional expansion has numerous advantages such as: (i)

* Corresponding author.
E-mail addresses: satchi.lapa@gmail.com (S. Dehuri), sbcho@cs.yonsei.ac.kr (S.-B. Cho).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.11.090

reducing or removing the amplification of noisy/irrelevant features, (ii) minimizing the complexity of the architecture, (iii) minimizing the computational load of training the network, etc. Another important point is no matter how intelligent the FLNN is, it will fail to predict the unknown sample if it is applied to low quality data. Hence, to improve the capability of the FLNN for accurate classification and to make more insight in the nature of the problem we rely on the hybridization of genetic algorithms and FLNN. Moreover, genetic selection is taken up due to the intractable nature and plagued by host of local minima in the exponential search space. This method not only has practical time complexity but also achieves good performance.

Over the last years, the machine learning and data mining community has become increasingly alert the need for statistical validation of the results. This can be ascribed to the maturity of the area, increasing the number of real-life applications and the availability of open algorithmic frameworks that make it easy to develop new algorithms or modify the existing methods, and compare them among themselves. In this paper, we used a set of simple, yet safe, robust and non-parametric tests for statistical comparisons of newly proposed method with FLNN and RBFN classifiers over multiple datasets. The reason being chosen these two classifiers is that FLNN is flat net without feature selection and has demonstrated well in the classification task of data mining (Misra & Dehuri, 2007). Light (1992), Powell (1992) has already been proved that RBFN can approximate arbitrarily well any decision boundary if a sufficient number of radial basis function units are given.

The rest of this paper is organized as follows. In Section 2, we discuss the background materials and related works. Section 3 provides the HFLNN for classification with optimal set of features. The robustness of the proposed method is illustrated in Section 4 by a mathematical analysis based on the stability and convergence analysis. In Section 5 we have presented the experimental studies and a parametric and non-parametric statistical comparative performance with other classifiers such as RBFN and FLNN trained by back propagation learning. Section 6 concludes the paper.

2. Background and related work

Adaptation of GAs in NNs has already proven a sound theoretical and empirical results from three aspects such as: GAs for optimizing weights in NNs, feature/instance selection in NNs (the objective is to reduce the size and dimension of the training set), and to design the structure of the network. However, to the best of our knowledge not a single attempt has been made for genetically select the most relevant features for functional expansion which in turns can be fed as the input of the FLNN for classification.

In this section we will discuss the basic background material required for a clear notion of the proposed method and some of the related work.

2.1. Genetic algorithms based feature selection in NNs

GAs (Goldberg, 1989) are stochastic search algorithms characterized by the fact that a number P of potential solutions (called individuals $ch_i \in \aleph$, where $\aleph$ represents the space of all possible individuals) of the optimization problem simultaneously sample the search space. This population $\Omega = \{ch_1, ch_2, \ldots, ch_P\}$ is modified according to the natural evolutionary process: after *initialization*, *selection* $S{:}\Omega \rightarrow ch_i$ and recombination $R{:}\Omega \rightarrow ch_i$ are executed in a loop until some termination criterion is reached. Each run of the loop is called a generation and $\Omega(t)$ denotes the population at generation t.

The selection operator is intended to improve the average quality of the population by giving fitter individuals a higher probability to be copied into the next generation. Selection thereby focuses on the search of promising regions in the search space. The quality of an individual is measured by a fitness function $f : \Omega \rightarrow \Re$. Recombination and mutation change the genetic material in the population in order to obtain new points in the search space.

Besides searching for the weights topology determination, genetic algorithms may be used to select relevant features that are input to the NNs for classification. The training samples may contain features that are irrelevant, noisy or redundant, but it is not known a priori which features are relevant. Avoiding these features is desirable not only because they increase the size of the network and the training time, but also they may reduce the accuracy of the network. Further, among the different categories of feature selection algorithms the GAs is a rather recent development. GA based feature selection is very essential because of the following reasons. Suppose there are m features in the data being mined. Then the total number of candidate feature subsets is 2^m that are the size of search space of the feature selection grows exponentially with the number of features.

The pioneering work by Siedlecki and Skalansky (1988) demonstrated evidence for the superiority of GA compared to representative classical algorithms. Subsequently many literatures were published that have shown advantages of GAs for feature selection in NNs (Brill, Brown, & Martin, 1990; Brotherton & Simpson, 1995; Yang & Honavar, 1998; Ozdemir et al., 2001).

2.2. Functional link neural networks for classification

Classification in the context of data mining is learning a function f that maps (classifies) a data item (i.e., called facts F) into one of several predefined classes. In real-life learning a function means that it has the capability to generate non-linear decision boundaries. It is known that using a number of hyper-planes one can approximate any nonlinear surface. Hence, the problem of classification can be viewed as searching for a number of linear surfaces that can appropriately model the class boundaries while providing minimum number of misclassified data points. Multiple layer neural networks can mostly be used for solving complex classification problems. However, depending on the complexities of the problems, the number of layers and number of neurons in the hidden layer need to be changed. As the number of layers and the number of neurons in the hidden layer increases, training the model becomes further complex. Very often different algorithms fail to train the model for a given problem set.

To overcome the complexities associated with multi-layer neural network, a single layer neural network can be considered as an alternative approach. But the single layer neural network being linear in nature very often fails to solve the complex nonlinear problems. The classification task in data mining is highly nonlinear in nature. Therefore for solving such problems in single layer feed forward artificial neural network is almost an impossible task.

In order to bridge the gap between the linearity in the single layer neural network and the highly complex and computationally intensive multi-layer neural networks, the FLNN architecture with back propagation learning for classifications was proposed (Misra & Dehuri, 2007). In contrast to the linear weighting of the input pattern produced by the linear links of artificial neural network, the functional link acts on an element of a pattern or on the entire pattern itself by generating a set of linearly independent functions, then evaluating these functions with the pattern as the argument. Thus class separability is possible in the enhanced feature space. The FLNN obtains the solution for W iteratively using BP algorithm based on all the training samples.

Learning of a FLNN may be considered as approximating or interpolating a continuous multivariate function $\phi(X)$ by an approximating function $\phi_W(X)$. In FLNN architecture, a set of basis

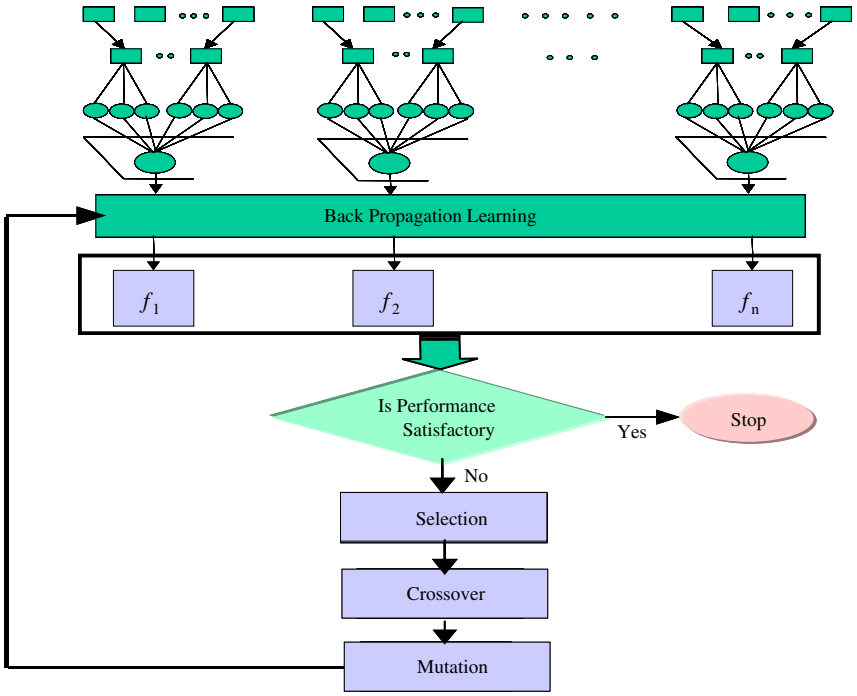

Fig. 1. Topological structure of the HFLNN.

functions Ψ, and a fixed number of weight parameters W are used to represent $\phi_W(X)$. With a specific choice of a set of basis functions Ψ, the problem is then to find the weight parameters W that provides the best possible approximation of ϕ on the set of input–output samples. This can be achieved by iteratively updating W. The interested reader about the detailed theory of FLNN can refer to (Misra & Dehuri, 2007; Pao, 1992; Pao & Philips, 1995; Pao & Takefuji, 1992).

Let k training patterns be applied to the FLNN and can be denoted by $\langle X_i{:}Y_i\rangle, 1 \leqslant i \leqslant k$ and let the weight matrix be W. At the ith instance $1 \leqslant i \leqslant k$, the Q-dimensional input pattern and the FLNN output are given by $X_i = \langle x_{i1}, x_{i2}, \ldots, x_{iQ}\rangle, 1 \leqslant i \leqslant k$ and $\widehat{Y}_i = [\hat{y}_i]$, respectively. Its corresponding target pattern is represented by $Y_i = [y_i], 1 \leqslant i \leqslant k$. Hence $\forall i, X = [X_1, X_2, \ldots, X_k]^T$. The augmented matrix of Q-dimensional input pattern and the FLNN output are given by:

$$\langle X : \widehat{Y}\rangle = \begin{pmatrix} x_{11} & x_{12} & \cdot & x_{1Q} : \hat{y}_1 \\ x_{21} & x_{22} & \cdot & x_{2Q} : \hat{y}_2 \\ \cdot & \cdot & \cdot & \cdot : \cdot \\ x_{k1} & x_{k2} & \cdot & x_{kQ} : \hat{y}_k \end{pmatrix}$$

As the dimension of the input pattern is increased from Q to Q' by a set of basis functions ψ, given by $\psi(X_i) = [\psi_1(x_{i1}), \psi_2(x_{i1}), \ldots, \psi_1(x_{i2}), \psi_2(x_{i2}), \ldots, \psi_1(x_{iQ}), \psi_2(x_{iQ}), \ldots] = [\psi_1(x_{i1}), \psi_2(x_{i2}), \ldots, \psi_{Q'}(x_{iQ})]$. The $k \times Q'$ dimensional weight matrix is given by $W = [W_1, W_2, \ldots, W_k]^T$, where W_i is the weight vector associated with the ith output and is given by $W_i = [w_{i1}, w_{i2}, \ldots, w_{iQ'}]$. The ith output of the FLANN is given by $\hat{y}_i(t) = \varphi\left(\sum_{j=1}^{Q'} \psi_j(x_{ij}) \cdot w_{ij}\right) = \varphi(W_i \cdot \psi^T(X_i)) \ \forall i$. The error associated with the ith output is given by $e_i(t) = y_i(t) - \hat{y}_i(t)$. Using the back propagation learning, weights of the FLNN can be updated as:

$$w_{ij}(t+1) = w_{ij}(t) + \mu \cdot \Delta(t), \quad \Delta(t) = \delta(t) \cdot [\psi(X_i)], \tag{1}$$

where $\delta(t) = [\delta_1(t), \delta_2(t), \ldots, \delta_k(t)], \delta_i(t) = \left(1 - \hat{y}_i^2(t)\right) \cdot e_i(t), \mu$ is known as the learning parameter.

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 is very high and further increasing the dimensionality to a very large extent may not be an appropriate choice. So, it is advisable and also a new research direction to choose a small set of alternative functions, which can map the function to the desired extent with an output of significant improvement.

Following the success of evolutionary multi-layer neural networks, various researchers have undertaken research in this arena. Haring and Kok (1995) use genetic algorithms and genetic programming for the determination of functional links. The functional links mainly based on polynomials and expression tree with basic atomic functions. They have tested on few artificial datasets and no comparison with competitive methods made. Sierra, Macias, and Corbacho (2001) use genetic algorithms for evolving polynomial functional inputs to tackle the combinatorial nature of the polynomial input terms for FLNN, which obviously precludes the exhaustive search for subsets of polynomial inputs. However, the method might be caught with the permutation problems, blurring of noisy features, etc.

Algorithms more or less related are amongst others: EAs to train the weights of the NNs, reduce the size of the training set by selecting the most relevant features and to design the topology of the NNs. For an extensive overview of the combinations of EAs and NNs see (Yao, 1999) and an empirical comparison of combinations of EAs and NNs for classification refer (Cant-Paz & Kamath, 2005).

The motivation behind the proposed method is that even though multi-layer neural networks can solve equal complex problems but it does not give much insight in the nature of the problem it solves. Additionally, there is a risk that NNs evolved by GAs will be so complex that interpretation is even more difficult than for regular multi-layer networks.

3. The proposed method

The proposed method (HFLNN) is a single hidden layer (i.e., it consists of the expanded units or known as higher order units)

artificial neural network (ANN) with genetically optimized features. It has the capability of generating complex decision regions by non-linear enhancement of hidden nodes referred to as functional links. Fig. 1 shows the topological structure of the method. The proposed method is characterized by a set of FLNN with a different subset of features. Then FLNN can approximate the function $\phi(X)$ by constructing the interpolating polynomial $\phi_W(X)$ with a specific choice of a set of basis functions ψ and a fixed set of optimum W. This is the idea behind a FLNN and higher order neural networks (HONs). FLNN are HONs without hidden layer.

In general, the basis function ψ is constructed as a function of the original attributes such as polynomial up to certain degree, trigonometric functions, exponential functions, and rational functions. The proposed method considered a set of trigonometric functions described in Section 3.1 and one linear function. The justification for choosing the trigonometric functions is given in Section 3.6.

Let us see the criterion of how best we can approximate the function out of the constructed one. The interpolating polynomial $\phi_W(X)$ can be defined as follows.

Definition. A polynomial $\phi_W(X)$ is called an interpolating polynomial with a set ψ and W if the values $\phi_W(X)$ and/or its certain order derivatives coincides with $\phi(X)$ in as many points as possible.

Let us assume that the function $\phi(X)$ is continuous in the given domain of interest. Then the following theorem shows the criterion of the best approximation.

Theorem (*Best approximations*). *For any $\phi(X)$, given any $\varepsilon > 0$,$\exists a$ $n = n(\varepsilon)$ an interpolating polynomial $\phi_W(X)$ such that $|\phi(X)-\phi_W(X)| \prec \varepsilon, \forall X$.*

Therefore, with a specific choice of ψ (an informal mathematical argument is given in Section 3.6 to show how good a set of basis functions for achieving predictive accuracy), the problem is then to find the weight parameters W that provides the best approximation of ϕ on the set of instances belongs to the given domain of interest.

The initial input of the network is same as the number of input variables of the data domain. Let Q be the number of original features of the data domain. The number of features selected to become a chromosome of the genetic population is q, $q \leqslant Q$. The q varies from chromosomes to chromosomes of the genetic population (i.e., $1 \leqslant q \leqslant Q$). For simplicity, let us see how a single chromosome with q features is working cooperatively for the proposed method.

3.1. Description of the basis functions

The general trigonometric function for mapping the q feature from one form to another form of higher dimension is considered as a set of promising basis functions. However, one can use a function that is very close to the underlying distribution of the data but it requires some prior domain knowledge. In this work we are taking five functions out of which four are trigonometric and one is linear (i.e., to keep the original form of the feature value intact). Out of the four trigonometric functions-two are *sine* and two are *cosine* functions. In the case of trigonometric functions the domain is feature values and range is a real number lies between $[-1,1]$. It can be written as

$$f : X \to R^{[-1,1]\cup\{x\}}, \tag{2}$$

where $X = \{x_{i1},x_{i2},\ldots,x_{iq}\}$, and q is known as the feature subset.

In general let us take $\psi_1,\psi_2,\ldots,\psi_N$ be the number of functions used to expand each feature value of the pattern. Therefore, each input pattern can now be expressed as

$$\begin{aligned}\vec{x}_i &= \{x_{i1},x_{i2},\ldots,x_{iq}\}\\ &\to \{\{\psi_1(x_{i1}),\psi_2(x_{i1}),\ldots,\psi_N(x_{i1})\},\ldots,\{\psi_1(x_{iq}),\psi_2(x_{iq}),\ldots,\psi_N(x_{iq})\}\}\\ &= \{\{y_{11},y_{21},\ldots,y_{N1}\},\ldots,\{y_{1q},y_{2q},\ldots,y_{Nq}\}\}\end{aligned} \tag{3}$$

In the present context, the $\vec{x}_i$ can be written as

$$\vec{x}_i = \begin{Bmatrix} \langle x_{i1},\sin\pi x_{i1},\sin 2\pi x_{i1},\cos\pi x_{i1},\cos 2\pi x_{i1}\rangle, & \langle x_{i2},\sin\pi x_{i2},\sin 2\pi x_{i2},\cos\pi x_{i2},\cos 2\pi x_{i2}\rangle \\ \ldots & \langle x_{iq},\sin\pi x_{iq},\sin 2\pi x_{iq},\cos\pi x_{iq},\cos 2\pi x_{iq}\rangle \end{Bmatrix}. \tag{4}$$

The set of basis functions has the following characteristics:

(1) $\psi_1 = x_{ij}, \forall i,j$,
(2) $\psi_k, 2 \leqslant k \leqslant N$ is linearly independent i.e., if $\sum_{k=2}^{N} w_k\phi_k = 0$, *then* $w_k = 0$, $\forall k$, *and*
(3) $\sup_N[\sum_{k=2}^{N}\|\phi_k\|^2]^{1/2} \prec \infty, N = 5$.

3.2. Chromosome with a single instance

In this section we will discuss the learning mechanism of a single chromosome based FLANN. Let $\psi = \{\psi_i\}_{i=1}^{N}$ be a set of basis functions to be considered for a FLANN based on the jth chromosome, $1 \leqslant j \leqslant P$, where P is the size of the population. Thus the FLANN consists of N basis functions $\langle \psi_1,\psi_2,\ldots,\psi_N\rangle \in \psi$ with the following input–output relationship for the lth pattern.

$$y = \phi(S_l), \quad S_l = \sum_{j=1}^{q}\left(\sum_{i=1}^{N}\phi_i(x_{lj}) \cdot w_{ji}\right), \tag{5}$$

where $X_l = [x_{l1},x_{l2},\ldots,x_{lq}]^T$ is the lth input pattern vector, $\hat{y}_l$ is the corresponding class label and $W_l = [w_{11},w_{12},\ldots,w_{qN}]$ is the weight vector associated with the lth output of the FLANN. Further, Eq. (8) can be written as

$$y = U(W\psi^T), \tag{6}$$

where $(U)_{1\times q}$ is a q-dimensional unit vector, $(W)_{qN}$ is a weight vector and $(\psi)_{1\times N}$ is an N-dimensional basis functions.

Note that $\psi = [\psi_1,\psi_2,\ldots,\psi_N]$ can be applicable for all attributes of the pattern vector X: for each row vector of W, there is a different domain for the set of basis functions. The criterion of interest for us to optimize is $|y-(U(W\psi^T))| \prec \varepsilon$, $\forall X$.

3.3. Chromosome with multiple instances

Suppose that there are k input–output training pattern pairs to be trained by the FLANN. Let the input pattern vector X be of dimension q and for ease of understanding, let the output y be a scalar. Each of the input patterns is passed through a functional expansion block producing a corresponding $(|dN|)$-dimensional, $(|dN|) > N$ expanded vector. In this case the dimension of the weight matrix is of $1 \times (|dN|)$ and hence, the individual weights are represented by $W = \langle w_{11},w_{12},\ldots,w_{dN}\rangle$. The linear weight is

passed through the sigmoidal non-linear function ϕ to produce the output y.

The FLANN with a set of basis functions $\{\psi\}_{i=1}^{N}$ attempts to approximate the non-linear function $\phi: A \subset R^n \to R$ using a set of k training patterns. Considering all k training patterns, input–output relationship may be expressed as

$$\psi \cdot W^T = Y^T, \tag{7}$$

where ψ,W and Y are the $k \times (|dN|)$,$1 \times (|dN|)$, and $1 \times k$ dimensional basis, weight and output vector, respectively. The corresponding matrix is as follows:

$$\psi = \begin{pmatrix} \psi_{1,11} & \psi_{1,12} & \cdot & \psi_{1,dN} \\ \psi_{2,11} & \psi_{2,12} & \cdot & \psi_{2,dN} \\ \cdot & \cdot & \cdot & \cdot \\ \psi_{k,11} & \psi_{k,12} & \cdot & \psi_{k,dN} \end{pmatrix},$$

$W = \langle w_{11}, w_{12}, \ldots, w_{dN} \rangle$, and $Y = \langle y_1, y_2, \ldots, y_k \rangle$.

Thus from Eq. (7) it is evident that finding the weights of the FLANN requires the solution of k simultaneous equations. Now depending upon the value of k and $(|dN|)$ three cases arise.

Case I: $k = (|dN|)$. If $\det(\psi) \neq 0$, then the solution for the weight vector is $W^T = \psi^{-1}Y$.

Case II: $k > (|dN|)$. This is a situation of under-determined problem and in that we may get a unique solution or may not. However, hoping for a unique solution one can select $(|dN|)$ row vectors from ψ matrix to obtain a matrix ψ_r and then compute the weight vector $W^T = \psi_r^{-1}Y$.

Case III: $k < (|dN|)$. The matrix ψ may be partitioned to obtain a matrix of ψ_f of dimension $k \times k$. Let W be reduced to W_f using any of the heuristic technique. If $\det(\psi_f) \neq 0$, then the weight vector can be computed by $W_f^T = \psi_f^{-1}Y$. The rest of the weight values are computed iteratively by the FLNN using training algorithm to be described below.

3.4. Training algorithm

Let k training patterns be presented to the network sequentially denoted by $\langle X_k, y_k \rangle$ and the weight vector of the network be denoted as $W(t)$, where t is the iteration. Referring to Eq. (2) the lth output of the FLNN at time t is

$$y_l(t) = \wp\left(\sum_{j=1}^{q}\left(\sum_{i=1}^{N}\phi_i(x_{ij}) \cdot w_{ji}\right)\right), \tag{8}$$

where $\psi = \{\psi_1, \psi_2, \ldots, \psi_N\}$, $\forall j$. Let the corresponding error be $e_k(t) = \hat{y}_k - y_k(t)$. Using the back propagation algorithm for a single layer, the update rule for all the weights of the FLANN is given by

$$W(t+1) = W(t) + \vartheta \partial_l(t) \psi_l(t), \tag{9}$$

where ϑ is a small constant and $\partial_l(t) = (1 - y_l(t))^2 e_l(t)$.

In other words the network has the ability to learn through back propagation learning. The training requires a set of training data, i.e., a series of input and associated output vectors. During the training, the network is repeatedly presented with the training data and the weights adjusted by back propagation learning from time to time till the following criterion function is optimized

$$E(t) = \sum_{i=1}^{k} e_i(t), \tag{10}$$

That means our goal must be to minimize Eq. (5) by gradient decent approach until $E \leqslant \varepsilon$.

Since we know all the aforesaid discussion is based on the single chromosome with a subset of attributes that can represent a FLANN therefore the survival of the FLANN and consequently the fitness of the chromosome is calculated by the following pseudocode.

```
fitness_Δ()
  {
  REPEAT
    {
    FOR l = 1 to k
      FOR each active bit of the chromosome
      Compute the matrix ψ_{1×dN}.
    END FOR
    COMPUTE
    y_l(t) = ℘( Σ_{j=1}^{q} ( Σ_{i=1}^{N} φ_i(x_lj).w_ji ) )
    COMPUTE ERROR
    e_l(t) = ŷ_l − y_l(t)
    UPDATE WEIGHT VECTOR
    W(t + 1) = W(t) + ϑ∂_l(t)ψ_l(t)
    E = E + e_l(t)
  END FOR
  }UNTIL(|E(t) − E(t + 1)| ⩽ ε)
}
```

The fitness of the ith chromosome ch_i which represents the ith FLNN can be defined as follows:

$$f_c(ch_i) = fitness\_\Delta(ch_i) - P_t(ch_i^q), \tag{11}$$

where ch_i^q is the corresponding feature subset of ch_i, and P_t is the penalty imposed on chromosomes breaking this constraint. The penalty P_t is measured by

$$P_t(ch_i^q) = (||ch_i^q| - s_q|) \times p_c, \tag{12}$$

where p_c is the penalty coefficient and s_q is constraint to force a feature subset to satisfy the given subset size requirement. The pseudocode for computing the fitness of a chromosome is written as follows:

```
Input: fitness_Δ(); output: fitness of ch_i
fitness_∇()
  {
    COMPUTE
      p_t(ch_i^q);
  EVALUATE
    fit_ch_i = fitness_Δ(ch_i) − p_t(ch_i^q);
  RETURN
    fit_ch_i;
  }
```

The value of penalty p_t is computed by the following pseudocode.

```
p_t(ch_i^q)
{
  COMPUTE PENALTY
    PT = (||ch_i^q| − s_q|) × p_c;
  RETURN
    PT;
}
```

This process is repeated for all chromosomes of the GA and then based on the performance each chromosome is assigned a fitness value. Using that fitness value the usual process of GA is executed until some good topology with high predictive accuracy is obtained.

3.5. Justification of trigonometric functions

The proposed structure with functional expansion using trigonometric functions has the following advantages:

- The trigonometric basis function was chosen here because this basis forms a more compact representation than other possible functions like Gaussian and orthogonal polynomials (e.g., Legendre, Chebyshev, etc). In addition the *sin* and *cos* functions can be computed quickly.
- Besides trigonometric functions, other basis is possible and may be chosen to be optimally useful for a particular class of problems or perhaps to maximize the performance of the classifier.
- It has been noted that if appropriate trigonometric polynomial is used for function expansion, the weight solution will approximate the terms in the multi-dimensional Fourier series decomposition version of the desired response function (Patra, 1999; Widrow & Lehr, 1990). In case of the proposed method the link acts on an element of a sample or on the entire sample itself by generating a set of independent functions. Then these functions are evaluated with the sample as the arguments. The functions are chosen as a subset of a complete set of orthonormal basis functions spanning an n-dimensional representation space, such as $sin\pi x$, $cos\pi x$, $sin2\pi x$, $cos2\pi x$ and so on. The net effect is to map the input sample from low to high-dimensional spaces. However, when the outer product terms were used in combination with the functional expansion good results were obtained in the case of learning the network.

3.6. Mathematical arguments

As we have already discussed above the learning of the proposed method can be considered as approximating or interpolating a continuous, multivariate function $\phi(x)$ by an approximating function $\phi_W(x)$. Here an informal mathematical argument is given to support how good the chosen basis functions for the task of classification. In the proposed method a set of trigonometric basis functions ψ and a fixed number of weight parameters W are used to represent the best approximator of $\phi_W(x)$ on the set of input–output samples.

Let A be a compact and simply connected subset of $\Re^n$ and $\ell_m(A)$ be the subset of Lebesgue measurable functions $\phi : A \subset \Re^n \to \Re^m$ such that the *sup* norm of ϕ, denoted by $\|\phi\|_A$ is bounded, i.e., $\|\phi\|_A = \sup_{x\in A}|\phi(x)| \prec \infty$. The space of all continuous functions $\phi : A \to \Re^m$ which is a subset of $\ell_m(A)$, and is denoted by $C_m(A)$. It is often desirable to parameterize ϕ in terms of discriminant $\sum_{i,j} w_{ij} \cdot \psi_j(X_i)$ to identify a non-linear function $\phi : A \to \Re^m$, for the selected trigonometric basis functions $\{\psi_i \in \ell(A)\}$. By Stone–Weierstrass theorem, there exist many sets of such functions that can uniformly approximate ϕ by a discriminant, if $\phi(x)$ is a continuous function over a compact set.

3.7. High level algorithms for proposed method

The specification of the near optimal proposed architecture and related parameters can be obtained by both genetic algorithms and back propagation learning, as it is explained in the following. The proposed method starts with a set of P-tuple vector or chromosomes $\Omega(t) = \langle ch_1(t), ch_2(t), \ldots, ch_P(t)\rangle$ of P individual $ch_i(t), i = 1(1)P$. Each individual $ch_i(t)$ is defined as a q-tuple or chromosome of binary value.

$$ch_i(t) = \left\langle 0, 1, 1, 0, \ldots \underbrace{\ldots, 0, 1}_{q\text{-tuple}} \right\rangle$$

The feature value corresponding to each active bit of the chromosome $ch_i(t)$ is mapped to an N-dimensional basis vector and the resultant vector $ech_i(t)$ can be represented as

$$ech_i(t) = \langle \psi_{i,11}, \psi_{i,12}, \ldots, \psi_{i,dN}\rangle.$$

Hence it is a dN-tuple vector $\phi_{i,hk} \in \Re$ with alphabet $\Re$. For simplicity and further analysis we represent the binary value of a chromosome by $ch_i(t) = \langle \ell_{i1}, \ell_{i2}, \ldots, \ell_{id}\rangle$. The individual fitness to be maximized is given by $fitness\_\nabla(ch_i(t))$. New individuals are created by the use of genetic operators-recombination and mutation. The topological structure of a given chromosome and its corresponding FLNN is given in Fig. 2.

3.7.1. Selection

In the proposed method the selection mechanism is implemented as a random process. According to the fitness proportionate selection procedure each individual $ch_i(t), i = 1(1)P$, is assigned the selection probability

$$P_i(t) = fitness\_\nabla(ch_i(t)) / \sum_{1\leqslant j\leqslant P} fitness\_\nabla(ch_j(t)). \tag{13}$$

Based on Eq. (13), the new population $\Omega(t+1)$ is generated by successively selecting individuals. The actual selection is done by the following roulette wheel procedure.

```
rw_selection()
  {
    (1) Compute cumulative probability of ch_i(t),
    p_i = Σ_{j=1(1)i} P_j, i = 1(1)P.
    (2) Generate a random number r ∈ [0,P_p].
    (3) Select the ith chromosome such that
    p_{i-1} < r < p_i, where p_0 = 0.
  }
```

The genetic operators that are most commonly used in this article are recombination and mutation operator. Here the single point recombination is considered with a probability of C_p.

3.7.2. Recombination operator

The single point crossover operator

$$C_{\otimes}(ch_i(t), ch_j(t)) = \left\langle ch_i'(t), ch_j'(t)\right\rangle$$

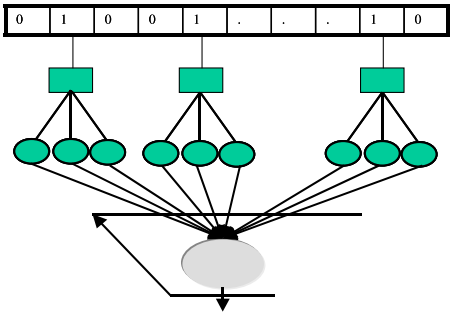

Fig. 2. Individual representation with its associated FLANN topology.

is applied to the selected individuals

$$ch_i(t) = \langle \ell_{i1}, \ell_{i2}, \ldots, \ell_{iq} \rangle$$
$$ch_j(t) = \langle \ell_{j1}, \ell_{j2}, \ldots, \ell_{jq} \rangle$$

creating $ch'_i(t)$ and $ch'_j(t)$ by the following mechanism. A random number r_N is choosen from {1, 2,...,q} with each number equally likely. Two new q tuples are formed from $ch_i(t)$ and $ch_j(t)$ by exchanging the alleles right from r_N leading to

$$ch'_i(t) = \langle \ell_{i1}, \ell_{i2}, \ldots, \ell_{ir_N} | \ell_{jr_N+1}, \ldots, \ell_{jq} \rangle$$

and

$$ch'_j(t) = \langle \ell_{j1}, \ell_{j2}, \ldots, \ell_{jr_N} | \ell_{ir_N+1}, \ldots, \ell_{iq} \rangle$$

3.7.3. Mutation operator

The mutation operator M_{flip} changes the alleles of the chromosomes $ch_i(t)$ and $ch_j(t)$ with small mutation probability.

The major steps of the proposed method in pseudocode format is as follows:

```
hflann_pseudocode()
  {
    SET t=0 and Initialize Population Ω(t);
    REPEAT
      {
      FOR all members of Ω(t)
        Evaluate fitness_∇();
      END FOR
      REPEAT
        {
        Select ch_i(t) and ch_j(t),1 ⩽ i,j ⩽ |P| from Ω(t) by
  rw_selection();
        Apply Crossover with C_p:
        ⟨ch'_i(t), ch'_j(t)⟩ = C_⊗(ch_i(t), ch_j(t));
        Apply Mutation with M_p:
        ch_i(t+1) = M_flip(ch'_i(t));
        and
        ch_j(t+1) = M_flip(ch'_j(t))
        }UNTIL <Adaptionof all P>
      }UNTIL <Termination Criterion>
  }
```

This algorithm is not only selecting the optimal set of features but also evolving a set of FLANN architecture. We can say this is a type of evolving FLANN. However, in this work we are not taking into account of optimizing the architecture from all aspects (i.e., weights). Hence, instead of a multi-objective function optimization we are only optimizing the uni-objective, i.e., known as predictive accuracy of the proposed method.

4. Stability and convergence analysis

The stability and convergence of the proposed method is based on the stochastic and a simple cluster mechanism of chromosomes of Ω. The convergence analysis is based on the stability analysis of the proposed method. Before detailed discussions there are few terminology needs to define.

- C^t_{ch}: the number of a cluster C in the population $\Omega(t)$ at generation t.
- C: the cluster of individuals selected randomly from $\Omega(t)$.
- $f(C)$: is the fitness of cluster C and
- $\overline{fitness\_\nabla}$: is the average fitness of $\Omega(t)$.

Under the fitness proportionate selection pressure without considering the genetic operators, the number of individuals of a cluster is expected to increase (or decrease) based on relative fitness:

$$C^{(t+1)}_{ch} = \frac{1}{\overline{fitness\_\nabla}}(C^{(t)}_{ch} \times f(C)). \tag{14}$$

Considering genetic operators and ignoring destruction ability the worst-case analysis with respect to cluster C yields:

$$C^{(t+1)}_{ch} \geqslant \frac{P_s}{\overline{fitness\_\nabla}}(C^{(t)}_{ch} \times f(C)), \tag{15}$$

where P_s is the probability of surviving genetic operations.

For the convergence analysis of the proposed method the objective is divided into two sets of stable individuals surviving and the unstable individuals perishing as quickly as possible. Let the population is divided into two sets, one of size $|s(t)|$ corresponding to the stable individuals represented by $CS^{(t)}_{ch}$, the other, $CU^{(t)}_{ch}$ consisting of the unstable members having size $|u(t)| = P - |s(t)|$. Let $f_S(t)$ be the fitness of $CS^{(t)}_{ch}$ and $f_U(t)$ be the fitness of $CU^{(t)}_{ch}$. Therefore, the average fitness of $\Omega(t)$ can be redefined as follows:

$$\overline{fitness\_\nabla(t)} = \frac{1}{P}(|s(t)|f_S(t) + |u(t)|f_U(t)) \tag{16}$$

Let us assume that the following relation holds $\forall t$ and $\eta > 1$

$$\frac{f_S(t)}{f_U(t)} = \eta. \tag{17}$$

The average fitness in terms of stable cluster statistics becomes

$$\overline{fitness\_\nabla(t)} = \frac{f_S(t)}{P}(|s(t)| + (|u(t)|/\eta)) \tag{18}$$

$$\Rightarrow \overline{fitness\_\nabla(t)} = \frac{f_S(t)}{P}\left(\frac{|s(t)|}{\eta}(\eta - 1) + \frac{P}{\eta}\right) \tag{19}$$

Therefore, the number of stable instances in cluster CS_{ch} from iteration t to $t+1$ is obtained by

$$CS^{(t+1)}_{ch} = \frac{1}{\overline{fitness\_\nabla}}(CS_{ch}(t)f_S(t)) \tag{20}$$

or we can write

$$f_S(t) = \frac{|s(t+1)| \times \overline{fitness\_\nabla}}{|s(t)|}, \tag{21}$$

and the size of the stable cluster is

$$|s(t+1)| = \frac{1}{\overline{fitness\_\nabla}}\langle |s(t)| \times f_S(t) \rangle \tag{22}$$

Now substituting (19) in Eq. (13) we get,

$$|s(t+1)| = \frac{|s(t)|}{\frac{1}{P}\left(\frac{s(t)}{\eta}(\eta - 1) + \frac{P}{\eta}\right)} \tag{23}$$

Eq. (22) can provide the expected number of stable individuals of the proposed approach at generation $(t+1)$ given η. For convenience, define $R_s(t)$ be the stable proportion of the population as:

$$R_s(t) = \frac{|s(t)|}{P}. \tag{24}$$

Therefore, dividing Eq. (22) by P, the expected proportion of the stable individuals in generation $(t+1)$ is given by

$$R_s(t+1) = \frac{|s(t+1)|}{P} = \frac{\eta|s(t)|}{P + |s(t)(\eta - 1)|}. \tag{25}$$

Substituting the value of $|s(t)|$ in Eq. (25) we get,

$$R_s(t+1) = \frac{\eta R_s(t)}{1+(\eta-1)R_s(t)}. \tag{26}$$

The following results are derived from the above relationships.

Lemma 1 (*Stability analysis*). *The expected proportion of stable individuals of cluster at generation t by assuming a non-zero initial proportion of stable cluster is given by*

$$R_s(t) = \frac{\eta^t R_s(0)}{(\eta^t-1)R_s(0)+1}. \tag{27}$$

Proof. We can prove this by induction. □

Basis $(t=1)$:

$$R_s(1) = \frac{\eta^1 R_s(0)}{(\eta^1-1)R_s(0)+1} = \frac{\eta R_s(0)}{(\eta-1)R_s(0)+1}.$$

Assumption. it holds for $t=m$

$$R_s(m) = \frac{\eta^m R_s(0)}{(\eta^m-1)R_s(0)+1}. \tag{28}$$

Show: it holds for $t = m+1$

By Eq. (26)

$$R_s(m+1) = \frac{\eta R_s(m)}{1+(\eta-1)R_s(m)}.$$

Substituting Eq. (28) and simplifying yields:

$$R_s(m+1) = \frac{\eta^{m+1} R_s(0)}{(\eta^{m+1}-1)R_s(0)+1}.$$

The following theorem for convergence of the proposed method then can be postulated.

Theorem 2. Convergence). *The proposed method under fitness proportionate selection is expected to converge to a population consisting of entirely stable cluster.*

Proof. By taking the limiting value of Eq. (27), and assuming $\eta \succ 1$ and $R_s(0) \succ 0$, then it shows that the proposed method is expected to converge to a population of stable clusters under fitness proportionate selection.

Therefore,

$$\underbrace{\lim}_{t\to\infty} R_s(t) \approx 1.$$

It is also important to note that from Eq. (26), for $\eta \succ 1$, and then the following relation holds.

$$R_s(t+1) \geqslant R_s(t) \qquad \square \tag{29}$$

5. Experimental studies

The performance of the proposed method was evaluated using a set of eleven public domain datasets from the University of California at Irvine (UCI) machine learning repository (Blake & Merz, 1998). In addition we have taken VOWEL dataset to show the performance of HFLNN to classify six overlapping vowel classes (Pal & Majumdar, 1977). We have compared the results of HFLNN with other competing classification methods such as radial basis function network (RBFN) and our previously proposed FLNN with back propagation learning.

This section is divided into five Subsections. In Section 5.1 the nature and characteristics of the dataset is described. Section 5.2 discusses the parameter set up required for the experiment. The comparative performance of the model is demonstrated in Section 5.3 with a discussion. The statistical analysis and asymptotic upper bound of the proposed method is presented in Sections 5.4 and 5.5, respectively.

5.1. Description of the datasets

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

IRIS Dataset: This is the most popular and simple classification dataset based on multivariate characteristics of a plant species (length and thickness of its petal and sepal) divided into three distinct classes (Iris Setosa, Iris Versicolor and Iris Virginica) of 50 instances each. One class is linearly separable from other two; the later are not linearly separable from each other. In a nutshell, it has 150 instances and 5 attributes. Among 5 attributes four attributes are predicting attributes and one is goal attribute. All the predicting attributes are real values.

WINE Dataset: These dataset are resulted from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. In classification context, this is a well-posed problem with well-behaved class structures. The total number of instances is 178 and it is distributed as 59 for class 1, 71 for class 2 and 48 for class 3. The number of attributes is 14 including class attribute and all 13 are continuous in nature. There are no missing attribute values in this dataset.

PIMA Indians Diabetes Database: This database is a collection of all female patients of at least 21 years old of PIMA Indian heritage. It contains 768 instances, 2 classes of positive and negative and 9 attributes including the class attribute. The attribute contains either integer or real values. There are no missing attribute values in the dataset.

BUPA Liver Disorders: This dataset is related to the diagnosis of liver disorders and created by BUPA Medical Research, Ltd. It consists of 345 records, 7 attributes including the class attribute. The class attribute is repeated with only two class values for entire database. The first 5 attributes are all blood tests, which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each record corresponds to a single male individual.

ECOLI: This dataset describes about the protein localization sites. It contains 336 instances, 7 predictive attributes with no missing values and one class attribute. The samples are distributed into 8 classes and the class distribution is highly unbalanced.

GLASS: The glass identification dataset contains 214 instances and 11 attributes (including an Id #) plus the class attribute (whose domain contains 6 values). All the attribute values are continuous and no one contains missing values.

VOWEL: This dataset consists of 871 patterns with 6 overlapping vowel classes and three input features. All entries are integers.

HOUSING: The Boston housing data concerns housing values in suburbs of Boston. There are 506 samples and 13 continuous attributes (including class attributes) and 1 binary valued attribute with no missing values.

LED7: The LED display dataset contain 7 attributes and user chooses the number of instances. No attribute contains missing values.

LYMPHOGRAPHY: This dataset contains 148 instances and 19 attributes (including the class attribute) with no missing values. The classes are distributed into 4 classes.

ZOO: This dataset contains 101 instances of zoo information. The number of attributes is 18 (animal name, 15 Boolean attribute, and 2 numeric attributes). There is no missing value in the domain of the attributes and it contains 7 types of zoo.

Table 1
Summary of the dataset used in simulation studies.

id	Dataset	Instances	Attribute	Classes	Missing values	Numeric/Categorical (Boolean)
Irs	Iris	150	4	3	No	4/0
Wne	Wine	178	13	3	No	13/0
Pma	Pima Indian diabetes	768	8	2	No	8/0
Bpa	Bupa liver disorder	345	6	2	No	6/0
Eli	Ecoli	336	7	8	No	7/0
Gls	Glass identification	214	9	6	No	9/0
Vwl	Vowel	871	3	6	No	3/0
Hng	Housing	506	13	5	No	11/2
Led7	Led7 display domain	300	7	10	No	0/7
Lym	Lymphography	148	18	4	No	0/18
Zoo	Zoo	101	16	7	No	2/15

Table 2
Average comparative performance of HFLNN, FLNN, and RBFN with a confidence interval level of $\alpha = 95\%$.

id	Folds	HFLNN	FLNN	RBFN
Irs	Training	0.9800 ± 0.0317	0.9667 ± 0.0406	0.3850 ± 0.1101
	Test	0.9733 ± 0.0365	0.9667 ± 0.0406	0.3850 ± 0.1101
Wne	Training	0.9944 ± 0.0155	0.9719 ± 0.0343	0.8539 ± 0.0734
	Test	0.9045 ± 0.0611	0.8877 ± 0.0656	0.7921 ± 0.0843
Pma	Training	0.8073 ± 0.0395	0.7956 ± 0.0403	0.7747 ± 0.0418
	Test	0.7214 ± 0.0448	0.7214 ± 0.0448	**0.7604 ± 0.0427**
Bpa	Training	0.7768 ± 0.0622	0.7797 ± 0.0619	0.7101 ± 0.0678
	Test	0.6928 ± 0.0687	0.6928 ± 0.0687	0.6695 ± 0.0701
Eli	Training	0.5517 ± 0.0752	0.4996 ± 0.0756	0.3118 ± 0.0700
	Test	0.5080 ± 0.0756	0.4731 ± 0.0755	0.2611 ± 0.0664
Gls	Training	0.6356 ± 0.0912	0.6075 ± 0.0925	0.4899 ± 0.0947
	Test	0.5151 ± 0.0947	0.5038 ± 0.0947	0.3464 ± 0.0902
Vwl	Training	0.4044 ± 0.0461	0.2793 ± 0.0422	0.2525 ± 0.0408
	Test	0.3820 ± 0.0456	0.2472 ± 0.0405	0.2432 ± 0.0403
Hng	Training	0.8221 ± 0.0471	0.7648 ± 0.0523	0.6719 ± 0.0579
	Test	0.7253 ± 0.0550	0.6976 ± 0.0566	0.6541 ± 0.0586
Led7	Training	0.3081 ± 0.0739	0.2242 ± 0.0667	0.2028 ± 0.0643
	Test	0.2753 ± 0.0715	0.1970 ± 0.0637	0.1657 ± 0.0595
Lym	Training	0.9730 ± 0.0369	0.9189 ± 0.0622	0.8513 ± 0.0811
	Test	0.7703 ± 0.0958	0.7432 ± 0.0995	0.7229 ± 0.1020
Zoo	Training	0.9904 ± 0.0270	0.9711 ± 0.0464	0.9615 ± 0.0533
	Test	0.8619 ± 0.0947	0.8516 ± 0.0976	0.8108 ± 0.1075

Table 1 presents a summary of the main characteristics of the databases that have been used in this study. The second column of this table gives the dataset name, while other columns indicate, respectively, the number of instances, the number of attributes, number of classes, missing values and whether numeric or categorical.

5.2. Parameter setup

For evaluating the proposed algorithm, the following user defined parameters and protocols related to the dataset need to be set beforehand.

A two fold cross-validation is carried out for all the dataset by randomly dividing the dataset into two parts (dataset1.dat and dataset2.dat). Each of these two sets was alternatively used either as training set or test set. To avoid over fitting, one may note that before generation stats, 20% of the instances in the training subset are marked as validation set for the use of BP algorithm; that is BP algorithm will use 80% of the original training subset for training the network and 20% for validation. Also the termination condition in our experiments occurs when the maximum number of generations reached; where one generation is equivalent to one complete pass through the training subset. The cross-validation set consists of the remaining 20% of the training subset will be used to estimate the generalization ability of the best result found during each generation.

It is also very important to set the optimal values of the following parameters to reduce the local optimality. The parameters are described as follows:

Population size: The size of the population denoted as $|P| = 50$ is fixed for all the datasets. We have chosen 50 to avoid under and over fit during the training. The larger the number of individuals the more computation time is required and the performance of the system will slow down.

Stop Criterion: The iteration is fixed to 1000 for all the datasets.

Length of the individuals is fixed to n, which is the number of input features. The value of n is determined corresponding to the number of features of the dataset. The probability for crossover is 0.7 and mutation is 0.02.

5.3. Comparative performance

The classification accuracy obtained from HFLNN for the above datasets were compared with the results obtained from FLNN with back propagation learning and radial basis function network (RBFN). Since the adaptation of GAs and NNs are stochastic multiple time experiments is required for measuring the performance and comparison with other methods. Therefore, we performed 30

Table 3
Average comparative performance of HFLNN, FLNN, and RBFN with a confidence interval level of $\alpha = 98\%$.

id	Folds	HFLNN	FLNN	RBFN
Irs	Training	0.9800 ± 0.0377	0.9667 ± 0.0483	0.3850 ± 0.1309
	Test	0.9733 ± 0.0434	0.9667 ± 0.0483	0.3850 ± 0.1309
Wne	Training	0.9944 ± 0.0184	0.9719 ± 0.0408	0.8539 ± 0.0872
	Test	0.9045 ± 0.0726	0.8877 ± 0.0780	0.7921 ± 0.1002
Pma	Training	0.8073 ± 0.0469	0.7956 ± 0.0479	0.7747 ± 0.0497
	Test	0.7214 ± 0.0533	0.7214 ± 0.0533	**0.7604 ± 0.0508**
Bpa	Training	0.7768 ± 0.0740	0.7797 ± 0.0736	0.7101 ± 0.0806
	Test	0.6928 ± 0.0817	0.6928 ± 0.0817	0.6695 ± 0.0833
Eli	Training	0.5517 ± 0.0894	0.4996 ± 0.0899	0.3118 ± 0.0833
	Test	0.5080 ± 0.0899	0.4731 ± 0.0755	0.2611 ± 0.0790
Gls	Training	0.6356 ± 0.1084	0.6075 ± 0.110	0.4899 ± 0.1126
	Test	0.5151 ± 0.1126	0.5038 ± 0.1126	0.3464 ± 0.1072
Vwl	Training	0.4044 ± 0.0548	0.2793 ± 0.0501	0.2525 ± 0.0485
	Test	0.3820 ± 0.0542	0.2472 ± 0.0481	0.2432 ± 0.0479
Hng	Training	0.8221 ± 0.0560	0.7648 ± 0.0621	0.6719 ± 0.0688
	Test	0.7253 ± 0.0654	0.6976 ± 0.0673	0.6541 ± 0.0697
Led7	Training	0.3081 ± 0.0878	0.2242 ± 0.0793	0.2028 ± 0.0765
	Test	0.2753 ± 0.0850	0.1970 ± 0.0757	0.1657 ± 0.0707
Lym	Training	0.9730 ± 0.0439	0.9189 ± 0.0739	0.8513 ± 0.0964
	Test	0.7703 ± 0.1139	0.7432 ± 0.1183	0.7229 ± 0.1212
Zoo	Training	0.9904 ± 0.0321	0.9711 ± 0.0552	0.9615 ± 0.0634
	Test	0.8619 ± 0.1126	0.8516 ± 0.1160	0.8108 ± 0.1278

independent runs and the average results are demonstrated in Tables 2 and 3 with a confidence level of $\alpha = 95\%$ and 98%. Bold typeface is used to highlight the results that are significantly different from the proposed method.

From Tables 2 and 3, a simple observation can be derived; by rejecting some of the original features leads to improved model in higher dimension in almost all the datasets except Bpa and Pma. As far as the test cases of Bpa are considered, it can be easily verified that the classification accuracy with the confidence level of 95% and 98% are same. However, a very small deviation can be observed with a confidence level of 95% and 98% in the case of FLNN and HFLNN using the training set of Bpa. In the case of Pma, RBFN performs better.

Table 4 illustrates a comparative performance of the proposed algorithm with FLNN and RBFN by using maximum classification performance value obtained in training and test sets. In the case of Iris dataset the training and test case performance of the HFLNN is same as FLNN, however the performance of HFLNN is demonstrating superior than RBFN. Similarly in the case of zoo dataset all algorithms are performing same results with 100% accuracy in training set, however in test cases the performance of the proposed method is significantly different. Considering the maximum training and testing performance of the methods, in all cases (except Irs and zoo) the proposed method draws a clear edge.

We plot the observed performances of these classification algorithms with respect to their classification performance for each test and training data set as shown in Fig. 3a and b. Numeral on the *X*-axis from 1 to 10 represents the data domain starting from iris to zoo. It clearly indicates that the classification accuracy of the same dataset could vary widely depending upon what kind of classification algorithm is applied to it.

Table 5 shows the reduction of functional inputs in percentage and the corresponding possible number of functional inputs for each of the domain obtained in HFLNN. On an average, the memory requirements and computational load of the number of functional inputs reduced to 56% which is a clear indication of harnessing the power of HFLNN in other tasks of data mining.

Fig. 4 shows the percentage of features selected. The *X*-axis represents the data domain and *Y*-axis represents the percentage of active bits in the optimal chromosome obtained during the training.

Table 4
Comparative performance w.r.t. maximum training/testing performance (Tr: training and Te: test).

id	Training/test results	HFLNN	FLNN	RBFN
Irs	Tr/Te	98.667/97.333	98.667/97.333	57.333/48.000
Wne	Tr/Te	100/91.011	97.753/93.258	86.517/82.022
Pma	Tr/Te	81.51/72.656	80.208/72.656	78.125/77.604
Bpa	Tr/Te	77.907/70.349	78.488/70.93	71.676/68.208
Eli	Tr/Te	59.829/54.701	52.137/52.137	38.462/27.434
Gls	Tr/Te	63.81/57.143	60.952/55.046	53.211/38.095
Vwl	Tr/Te	40.708/41.88	33.628/28.205	27.434/25.641
Hng	Tr/Te	85.375/77.075	79.842/71.542	70.356/66.088
Led7	Tr/Te	34.188/35.398	33.333/27.434	29.06/19.469
Lym	Tr/Te	97.297/78.378	94.595/77.027	86.486/75.676
Zoo	Tr/Te	100/87.755	100/85.714	100/84.615

Fig. 5 shows the classification performance of the proposed method by incorporating individual knowledge with $\tau = 0.01$ in $f(T_1) = \frac{|CA|\times|n| - \tau\times|T_1|}{|n|}$ and $f(T_2) = \frac{|CA|\times|n| - \tau\times|T_2|}{|n|}$ for training set 1 (T_1) and training set 2 (T_2) with respect to their corresponding active bits of the individual. Here we denote the classification accuracy by $|CA|$ and total number of input features by $|n|$.

5.4. Statistical analysis

5.4.1. Paired t-test

We have tested the proposed method with FLNN and RBFNusing the *t*-test individually for training and testing performance scores. In order to test the significance of our algorithm over to FLNN and RBFN we first construct the null hypothesis. The null hypothesis is that there is no difference between the average performance of proposed method vs. FLNN and RBFN. In comparing the proposed method with FLNN using paired *t*-test with a *t*_value of 0.3833 and the degree of freedom 20, we rejected the null hypothesis, as the calculated *t*_value is less than the tabulated value 2.09

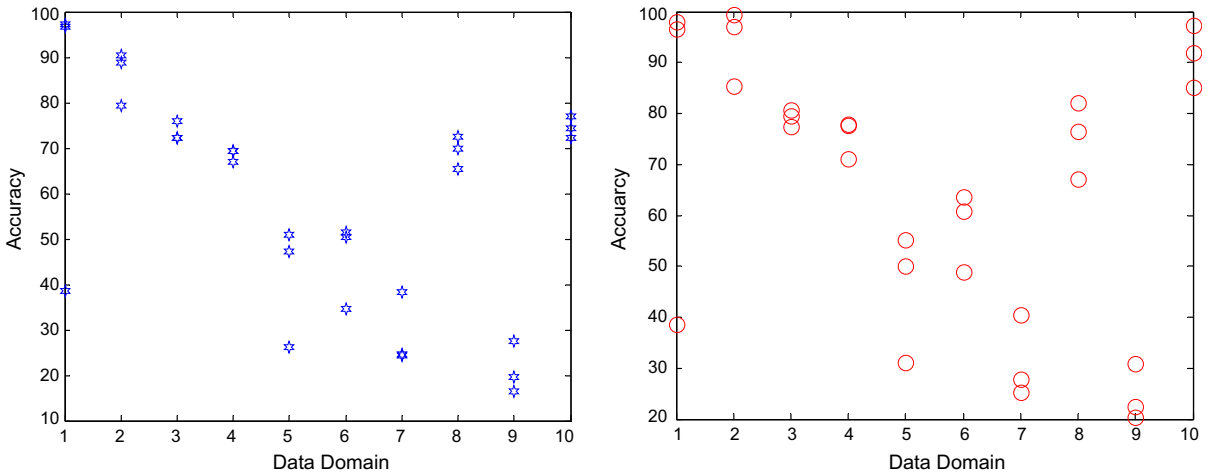

Fig. 3. Distribution of classification performance using HFLNN, FLNN, and RBFN (a) based on the test set and (b) training set.

Table 5
Reduction of functional inputs and the possible number of functional expansion.

id	Reduction of functional inputs in percentage	Possible number of functional inputs
Irs	75	20
Wne	46.1	65
Pma	75	40
Bpa	75	30
Eli	57.1	35
Gls	55.5	45
Vwl	33.3	15
Hng	76.9	65
Led7	57.1	35
Lym	38.9	90
Zoo	31.2	80

with the chosen level of significance 0.05. The comparative result of the proposed method with RBFN is also significantly different using the paired t-test with a calculated t-value of 1.4751 and degree of freedom 20.

5.4.2. Wilcoxon signed ranks test

Like paired t-test we will test the proposed method HFLNN with FLNN and RBFN separately because it can compare two algorithms at a time over multiple datasets. Here we are trying to reject the null hypothesis that both algorithms perform equally well. The ranks are assigned from the lowest to the highest absolute difference, and the equal differences are assigned average ranks. Tables 6 and 7 show the classification performance of HFLNN vs. FLNN and HFLNN vs. RBFN and their corresponding ranks considering the training set.

The sum of the ranks for positive difference is $R_{pos} = 65$ and the sum of the ranks for the negative difference is $R_{neg} = 1$. According to the table of exact critical values for the Wilcoxon's test, for a confidence level of $\alpha = 0.05$ and $N = 11$ datasets, the difference between the classifiers is significant if the smaller of the sums is equal or less than 11. We therefore reject the null hypothesis.

The sum of the ranks for positive difference is $R_{pos} = 66$ and the sum of the ranks for the negative difference is $R_{neg} = 0$. According to the table of exact critical values for the Wilcoxon's test, for a confidence level of $\alpha = 0.05$ and $N = 11$ datasets, the difference between the classifiers is significant if the smaller of the sums is equal or less than 11. We therefore reject the null hypothesis. Hence we can conclude that in both the cases the proposed algorithm is significantly different from 0.

5.5. Asymptotic computational complexity analysis

In this section we present an asymptotic computational complexity analysis of the proposed method. Step 1 of the algorithm is the initialization of the chromosomes and it takes $O(P \cdot Q)$ time, where P is the size of the population and Q is the feature space.

Next we have to consider the heart of the algorithms such as REPEAT loop. Let us first consider a single iteration of this loop,

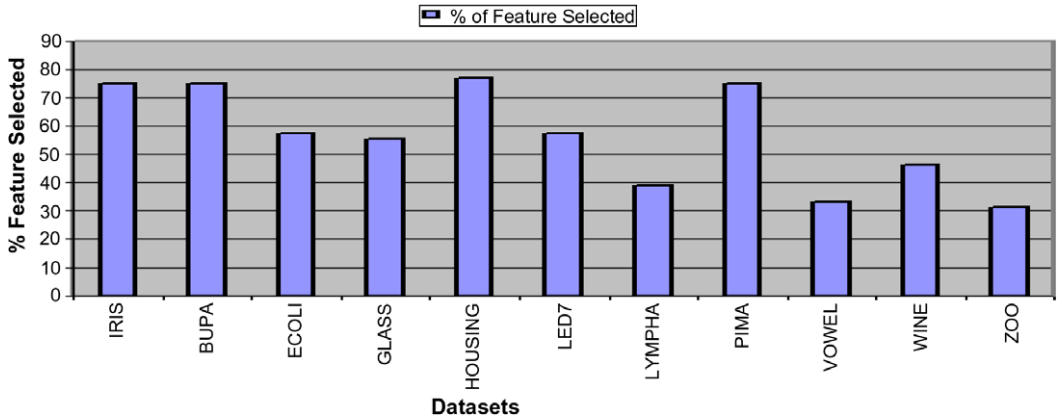

Fig. 4. Percentage of optimal set of selected features.

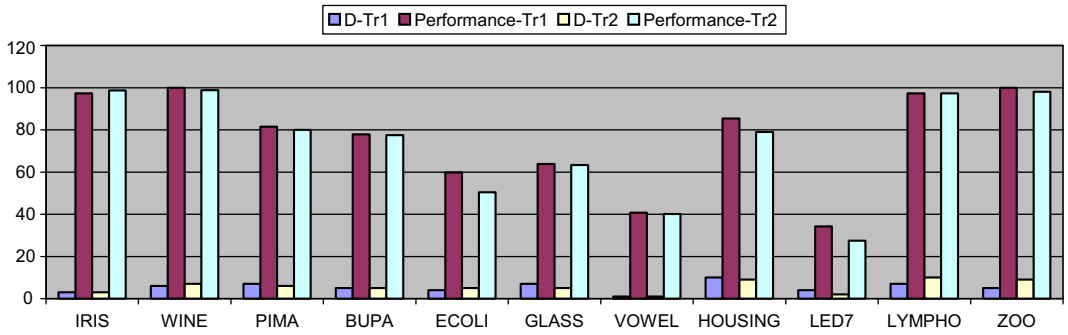

Fig. 5. Performance of the proposed method on Training set 1 and Training Set 2 with respect to the individual active bits.

Table 6
Performance score and ranks of HFLNN vs. FLNN.

id	HFLNN	FLNN	Difference	Rank
Irs	98.0001	96.6665	1.3336	3
Wne	99.4380	97.1910	2.2470	5
Pma	80.7290	79.5570	1.1700	2
Bpa	77.6820	77.9728	−0.2905	1
Eli	55.1670	49.9625	5.2045	7
Gls	63.5565	60.7510	2.8055	6
Vwl	40.4395	27.9250	12.5145	11
Hng	82.2130	76.4825	5.7305	9
Led7	30.8110	22.4185	8.3925	10
Lym	97.2970	91.8920	5.4050	8
Zoo	99.0385	97.1155	1.9230	4

Table 7
Performance Score and ranks of HFLNN vs. RBFN.

id	HFLNN	RBFN	Difference	Rank
Irs	98.0001	38.5000	59.5001	11
Wne	99.4380	85.3935	14.0445	6
Pma	80.7290	77.4740	3.2550	2
Bpa	77.6820	71.0125	6.6695	3
Eli	55.1670	31.1780	23.9890	10
Gls	63.5565	48.9865	14.5700	7
Vwl	40.4395	25.2555	15.1840	9
Hng	82.2130	67.1940	15.0190	8
Led7	30.8110	20.2820	10.5290	4
Lym	97.2970	85.1350	12.1620	5
Zoo	99.0385	96.1540	2.8845	1

and later on the entire REPEAT loop. Inside the outer one there is another REPEAT loop for adoptation of all chromosomes of P. The upper bound complexity of this loop requires $O(P)$. Similarly to compute the fitness of all P it requires $O(P)$. Hence ignoring the constant term the total time complexity for a single iteration of the outer REPEAT is equal to $O(P)$. Assuming we are considering for executing the algorithm for a fixed number of iterations T. Therefore after simplification the total computational complexity of the algorithm is $O(TP)$.

Note that we are given the asymptotic time complexity of the *hflann_pseudocode*( ) ignoring the time complexity of constituents algorithms. However, one can easily analyse the computational complexity of algorithms which are the constituents of *hflann_pseudocode*( ).

6. Conclusions

In this paper, we have evaluated the proposed method for the task of classification in data mining by giving an equal importance to the selection of optimal set of features and classification accuracy. The proposed model functionally maps the selected set of feature value from lower to higher dimension. The experimental studies demonstrated that the classification performance of the proposed model is promising. In almost all cases, the results obtained with the proposed method proved to be better than the best results found by its competitor like RBFN and FLNN with back propagation learning. Further, we empirically analyzed parametric (t-test) and non-parametric (Wilcoxon signed rank test) tests that can be used for comparing classifiers over multiple datasets. In addition to the generalization performance of HFLNN, the stability and convergence analysis shows how much changes in the training data and required parameters influence the proposed method. The convergence analysis reveals that convergence to the global optimum is not an inherent property of the HFLNN rather is a consequence of optimally choosing the parameters.

Although parsimonious models require less computational effort and often generalize better than overly complex models when facing novel data, the more immediate value of the compact HFLNN stems from the possibility for explaining why the networks are making specific decisions. Furthermore, whatever the facet of the original feature space, HFLNN select from the very beginning on small pertinent set of most favorable features. Therefore, the noisy features are not likely to haze it's functioning as would happen in other combinations of EAs with NNs and HONs, for which high dimension has proved to be curse. In this respect other applications definitely will get benefit where dimension is high.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (R01-2008-000-20801-0).

References

Blake, C. L., & Merz, C. J. (1998). UCI Repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Branke, J. (1995). Evolutionary algorithms in neural network design and training – A review. In Jarmo T. Alander (Ed.), *Proceedings of the first nordic workshop on genetic algorithms and their applications* (pp. 145–163). Vaasa, Finland.

Brill, F. Z., Brown, D. E., & Martin, W. N. (1990). *Genetic algorithms for feature selection for counterpropagation networks*. Tech. rep. IPC-TR-90-004. Charlottesville, VA: Univ. Virginia, Inst. Parallel Comput.

Brotherton, T. W., & Simpson, P. K. (1995). Dynamic feature set training of neural nets for classification. In J. R. McDonnell, R. G. Reynolds, & D. B. Fogel (Eds.), *Evolutionary programming IV* (pp. 83–94). Cambridge, MA: MIT Press.

Cant-Paz, E., & Kamath, C. (2005). An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, 35*(5), 915–927.

Caudell, T. P., & Dolan, C. P. (1989). Parametric connectivity: Training of constrained networks using genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the third international conference on genetic algorithms* (pp. 370–374). San Mateo, CA.

Fayyad, U. M., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, & P. Smyth (Eds.), *Advances in knowledge discovery and data mining* (pp. 1–34). Menlo Park, CA: AAAI Press.

Fogel, D. B., Fogel, L. J., & Porto, V. W. (1990). Evolving neural networks. *Biological Cybernetics, 63*, 487–493.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Morgan Kaufmann.

Haring, B., & Kok, J. N. (1995). Finding functional links for neural networks by evolutionary computation. In T. Van de Merckt, et al. (Eds.), *BENE-LEARN'95, Proceedings of the fifth Belgian–Dutch conference on machine learning* (pp. 71–78). Brussels, Belgium.

Kitano, H. (1990). Empirical studies on the speed of convergence of neural network training using genetic algorithms. In *Proceedings of the eighth national conference artificial intelligence* (pp. 789–795).

Kriegel, H.-P. et al. (2007). Future trends in data mining. *Data Mining and Knowledge Discovery, 15*(1), 87–97. Netherlands: Springer.

Light, W. A. (1992). Some aspects of radial basis function approximation. *Approximation Theory, Spline Functions Application, 356*, 163–190.

Miller, G. F., Todd, P. M., & Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In J. D. Schaffer (Eds.), *Proceedings of third international conference on genetic algorithms* (pp. 379–384). San Mateo, CA.

Misra, B. B., & Dehuri, S. (2007). Functional link artificial neural network for classification task in data mining. *Journal of Computer Science, 3*(12), 948–955.

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on artificial intelligence* (pp. 762–767). San Mateo, CA.

Oh, II-S., Lee, J.-S., & Moon, B.-R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(11), 1424–1437.

Ozdemir, M., Embrechts, M. J., Arciniegas, F., Breneman, C. M., Lockwood, L., & Bennett, K. P. (2001). Feature selection for in-silico drug design using genetic algorithms and neural networks. In *Proceedings of the IEEE mountain workshop soft computer industrial application* (pp. 53–57).

Pal, S. K., & Majumdar, D. D. (1977). Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Transactions on Systems Man and Cybernetics, 7*, 625–629.

Pao, Y.-H. et al. (1992). Neural-net computing and intelligent control systems. *International Journal of Control, 56*(2), 263–289.

Pao, Y. H., & Philips, S. M. (1995). The functional link net and learning optimal control. *Neurocomputing, 9*, 149–164.

Pao, Y. H., & Takefuji, Y. (1992). Functional link net computing: Theory, system, architecture and functionalities. *IEEE Computer*, 76–79.

Patra, J. C. et al. (1999). Identification of nonlinear dynamic systems using functional link artificial neural networks. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, 29*(2), 254–262.

Powell, M. J. D. (1992). Radial basis functions in 1990. *Advance Numerical Analysis, 2*, 105–210.

Seidlecki, W., & Skalansky, J. (1989). A note on genetic algorithms for large scale feature selection. *Pattern Recognition Letters, 10*, 335–347.

Siedlecki, W., & Skalansky, J. (1988). On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence, 2*(2), 197–220.

Sierra, A., Macias, J. A., & Corbacho, F. (2001). Evolution of functional link networks. *IEEE Transactions on Evolutionary Computation, 5*(1), 54–65.

Whitley, D., & Hanson, T. (1989). Optimizing neural networks using faster, more accurate genetic search. In J. D. Schaffer (Ed.), *Proceedings of the third international conference genetic algorithms* (pp. 391–397). San Mateo, CA.

Widrow, B., & Lehr, M. A. (1990). Thirty years of adaptive neural networks: Perceptron, madaline, and back-propagation. In *Proceedings of IEEE* (pp. 1415–1442, Vol. 78(9).

Yang, J. H., & Honavar, V. (1998). Feature subset selection using genetic algorithm. *IEEE Intelligent Systems, 13*(2), 44–49.

Yang, J., & Honavar, V. (1998). Feature subset selection using a genetic algorithm. *IEEE Intelligence Systems, 13*(2), 44–49.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE, 87*(9), 1423–1447.

Zhang, G. P. (2000). Neural networks for classification: A survey. *IEEE Transactions on Systems, Man, Cybernetics-Part C: Application and Reviews, 30*(4), 451–461.

Zhang, G. P. (2007). Avoiding pitfalls in neural network research. *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews, 37*(1), 3–16.