Hindawi Publishing Corporation
Advances in Artificial Neural Systems
Volume 2011, Article ID 107498, 10 pages
doi:10.1155/2011/107498

Research Article

A Novel Learning Scheme for Chebyshev Functional Link Neural Networks

Satchidananda Dehuri

Department of Information and Communication Technology, Fakir Mohan University, Vyasa Vihar, Balasore, Orissa 756019, India

Correspondence should be addressed to Satchidananda Dehuri, satchi.lapa@gmail.com

Received 11 January 2011; Revised 17 April 2011; Accepted 28 May 2011

Academic Editor: Giacomo Indiveri

A hybrid learning scheme (ePSO-BP) to train Chebyshev Functional Link Neural Network (CFLNN) for classification is presented. The proposed method is referred as hybrid CFLNN (HCFLNN). The HCFLNN is a type of feed-forward neural networks which have the ability to transform the nonlinear input space into higher dimensional-space where linear separability is possible. Moreover, the proposed HCFLNN combines the best attribute of particle swarm optimization (PSO), back propagation learning (BP learning), and functional link neural networks (FLNNs). The proposed method eliminates the need of hidden layer by expanding the input patterns using Chebyshev orthogonal polynomials. We have shown its effectiveness of classifying the unknown pattern using the publicly available datasets obtained from UCI repository. The computational results are then compared with functional link neural network (FLNN) with a generic basis functions, PSO-based FLNN, and EFLN. From the comparative study, we observed that the performance of the HCFLNN outperforms FLNN, PSO-based FLNN, and EFLN in terms of classification accuracy.

1. Introduction

In recent years, higher-order neural networks [1], particularly FLNN, have been widely used to classify nonlinearly separable patterns and can be viewed as a problem of approximating an arbitrary decision boundary. Broadly, artificial neural networks have become one of the most acceptable soft computing tools for approximating the decision boundaries of a classification problem [2, 3]. In fact, a multilayer perceptron (MLP) with a suitable architecture is capable of approximating virtually any function of interest [4]. This does not mean that finding such a network is easy. On the contrary, problems, such as local minima trapping, saturation, weight interference, initial weight dependence, and overfitting, make neural network training difficult.

An easy way to avoid these problems consists in removing the hidden layers. This may sound a little inconsiderate at first, since it is due to them that nonlinear input-output relationships can be captured. Encouragingly enough, the removing procedure can be executed without giving up nonlinearity, provided that the input layer is endowed with additional higher-order units [5, 6]. This is the idea behind higher-order neural networks (HONNs) [7] like functional link neural networks (FLNNs) [8], ridge polynomial neural networks (RPNNs) [1, 7], and so on. HONNs are simple in their architectures and require fewer number of weights to learn the underlying approximating polynomials. This potentially reduces the number of required training parameters. As a result, they can learn faster since each iteration of the training procedure takes less time. This makes them suitable for complex problem solving where the ability to retrain or adopt new data in real time is critical. Currently, there have been many algorithms used to train the functional link neural networks, such as back-propagation learning algorithm [2], genetic algorithm [9], particle swarm optimization [10], and so on. Back-propagation learning algorithms have their own limitations. However, we can advocate that if the search for the BP learning algorithms starts from the near optimum with a small tuning of the learning parameters, the searching results can be improved.

Genetic algorithms and particle swarm optimization can be used for training the FLNN to reduce the local optimality and speed up the convergence. But training using genetic algorithm is discouraging because of the following limitations: in the training process, it requires encoding and decoding operator which is commonly treated as a long-standing barrier of neural networks researchers. The important problem of applying genetic algorithms to train neural networks may be unsatisfactory because recombination operators incur several problems, such as competing conventions [11] and the epistasis effect [12]. For better performance, real coded genetic algorithms [13, 14] have been introduced. However, they generally employ random mutations, and, hence, still require lengthy local searches near a local optima. On the other hand, PSO has some attractive properties. It retains previous useful information, whereas GAs destroy the previous knowledge of the problems once the population changes. PSO encourages constructive cooperation and information sharing among particles, which enhances the search for a global optimal solution. Successful applications of PSO to some optimization problems such as function minimization [15, 16] and neural networks design [17, 18] have demonstrated its potential. It is considered to be capable to reduce the ill effect of the BP learning algorithm of neural networks, because it does not require gradient and differentiable information.

Unlike the GA, the PSO algorithm has no complicated operators such as cross-over and mutation. In the PSO algorithm, the potential solutions, called as particles, are obtained by flowing through the problem space by following the current optimum particles. Generally speaking, the PSO algorithm has a strong ability to find the most optimistic result, but it has a disadvantage of easily getting into a local optimum. After suitably modulating the parameters for the PSO algorithm, the rate of convergence can be speeded up, and the ability to find the global optimistic result can be enhanced. The PSO algorithm search is based on the orientation by tracing *pbest*, that is, each particle's best position in its history, and tracing *gbest* that is all particles best position in their history, it can rapidly arrive around the global optimum. However, because the PSO algorithm has several parameters to be adjusted by empirical approach, if these parameters are not appropriately set, search will proceed very slow near the global optimum. Hence, to cope up with this problem, we suggested a novel evolvable PSO (ePSO) and back propagation (BP) algorithm as a learning method of Chebyshev functional link neural network (CFLNN) for fine tuning of the connection weights.

1.1. Outline. The remainder of this paper is organized as follows. Some the recently proposed functional link neural networks (FLNNs) are reviewed in Section 2. Section 3 provides the detailed algorithm of HCFLNN for classification. In Section 4, we have presented experimental results with a comparative study. Section 5 concludes the article with a future research scope.

2. Functional Link Neural Networks

FLNNs are higher order neural networks without hidden units introduced by Klassen et al. [19] in 1988. Despite their linear nature, FLNNs can capture nonlinearly 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 [20].

In contrast to the linear weights of the input patterns 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 an argument. Thus, class separability is possible in the enhanced feature space. For a D-dimensional classification problem, there are $((D+r)!/D! \cdots r!)$ possible polynomials up to degree r that can be constructed. For most of the real life problems, this is too big number, even for degree 2, which obviously discourages us from achieving our goal. However, we can still resort to constructive and pruning algorithms in order to address this problem. In fact, Sierra et al. [21] have proposed a new algorithm for the evolution of functional link networks which makes use of a standard GAs [9] to evolve near minimal linear architectures. Moreover, the complexity of the algorithm still needs to be investigated.

However, the dimensionality of many problems is itself very high and further increasing the dimensionality to a very large extent that 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. FLNN with a trigonometric basis functions for classification, as proposed in [8], is obviously an example. Chebyshev FLNN is also another improvement in this direction, the detailed is discussed in Section 3. Some of the potential contributions in FLNNs and their success for application in variety of problems are given below.

Haring and Kok [22], has proposed an algorithm 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 and Pal [23] have proposed a FLNN and applied to the problem of channel equalization in a digital communication channel. It relies on BP-learning algorithm. Haring et al. [24] were presenting a different ways to select and transform features using evolutionary computation and show that this kind of selection of features is a special case of so-called functional links.

Dash et al. [25] have proposed a FLNN with trigonometric basis functions to forecast the short-term electric load. Panagiotopoulos et al. [26] have reported better results by applying FLNN for planning in an interactive environment between two systems: the challenger and the responder. Patra et al. [27] have proposed a FLNN with back-propagation learning for the identification of nonlinearly dynamic systems.

With the encouraging performance of FLNN [23, 27], Patra and van den Bos [28] further motivated and came up with another FLNN 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.

In [21], 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 the 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.

A Chebyshev functional link artificial neural networks have been proposed by Patra and Kot [29] for nonlinearly dynamic system identification. This is obviously another improvement in this direction and also a source of inspiration to further validate this method in other application domain. The proposed method is clearly an example. Singh and Srivastava [30] have estimated the degree of insecurity in a power system with a set of orthonormal trigonometric basis functions.

In [31], an evolutionary search of genetic type and multiobjective optimization such as accuracy and complexity of the FLNN in the Pareto sense is used to design a generalized FLNN with internal dynamics and applied to system identification.

Majhi and Shalabi [32] have applied FLNN for digital watermarking, their results show that FLNN has better performance than other algorithms in this line. In [33], a comparative performance of three artificial neural networks has been given for the detection and classification of gear faults. Authors reported that FLNN is comparatively better than others.

Misra and Dehuri [8] have used a FLNN for classification problem in data mining with a hope to get a compact classifier with less computational complexity and faster learning. Purwar et al. [34] have proposed a Chebyshev functional link neural network for system identification of unknown dynamic nonlinearly discrete-time systems. Weng et al. [35] have proposed a reduced decision feedback Chebyshev functional link artificial neural networks (RDF-CFLANN) for channel equalization.

Two simple modified FLANNs are proposed by Krishnaiah et al. [36] 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 [37], a FLANN with trigonometric polynomial functions is used in intelligent sensors for harsh environment that effectively linearizes the response characteristics, compensates for nonidealities, and calibrates automatically. Dehuri et al. [38] have proposed a novel strategy for feature selection using genetic algorithm and then used as the input in FLANN for classification.

With this discussion, we can conclude that a very few applications of HONNs have so far been made in classification task. Although theoretically this area is rich, but application specifically in classification is poor. Therefore, the proposed contribution can be another improvement in this direction.

3. Hybrid Chebyshev FLNN

3.1. Chebyshev Functional Link Neural Network. It is well known that the nonlinearly approximation of the Chebyshev orthogonal polynomial is very powerful by the approximation theory. Combining the characteristics of the FLNN and Chebyshev orthogonal polynomial the Chebyshev functional link neural network what we named as CFLNN is resulted. The proposed method utilizes the FLNN input-output pattern, the nonlinearly approximation capabilities of Chebyshev orthogonal polynomial, and the evolvable particle swarm optimization(ePSO)-BP learning scheme for classification.

The Chebyshev FLNN used in this paper is a single-layer neural network. The architecture consists of two parts, namely transformation part (i.e., from a low-dimensional feature space to high-dimensional feature space) and learning part. The transformation deals with the input feature vector to the hidden layer by approximate transformable method. The transformation is the functional expansion (FE) of the input pattern comprising of a finite set of Chebyshev polynomial. As a result, the Chebyshev polynomial basis can be viewed as a new input vector. The learning part uses the newly proposed ePSO-BP learning.

Alternatively, we can approximate a function by a polynomial of truncated power series. The power series expansion represents the function with a very small error near the point of expansion, but the error increases rapidly as we employ it at points farther away. The computational economy to be gained by Chebyshev series increases when the power series is slowly convergent. Therefore, Chebyshev series are frequently used for approximations to functions and are much more efficient than other power series of the same degree. Among orthogonal polynomials, the Chebyshev polynomials converge rapidly than expansion in other set of polynomials [8]. Moreover, Chebyshev polynomials are easier to compute than trigonometric polynomials. These interesting properties of Chebyshev polynomial motivated us to use CFLNN for approximation of decision boundaries in the feature space.

3.2. Evolvable Particle Swarm Optimization (ePSO). Evolvable particle swarm optimization (ePSO) is an improvement over the PSO [10]. PSO is a kind of stochastic algorithm to search for the best solution by simulating the movement and flocking of birds. The algorithm works by initializing a flock of birds randomly over the searching space, where every bird is called as a particle. These particles fly with a certain velocity and find the global best position after some iteration. At each iteration k, the ith particle is represented by a vector x_i^k in multidimensional space to characterize its position. The velocity v_i^k is used to characterize its velocity. Thus, PSO maintains a set of positions:

$$S = \left\{x_1^k, x_2^k, \ldots, x_N^k\right\} \tag{1}$$

and a set of corresponding velocities

$$V = \left\{v_1^k, v_2^k, \ldots, v_N^k\right\}. \tag{2}$$

Initially, the iteration counter $k = 0$, and the positions x_i^0 and their corresponding velocities v_i^0 $(i = 1, 2, \dots, N)$ are generated randomly from the search space Ω. Each particle changes its position x_i^k per iteration. The new position x_i^{k+1} of the ith particle $(i = 1, 2, \dots, N)$ is biased towards its best position p_i^k with minimized functional value $f(\cdot)$ referred to as personal best or *pbest*, found by the particle so far, and the very best position p_g^k, referred to as the global best or *gbest*, found by its companions. The *gbest* is the best position in the set

$$P = \left\{p_1^k, p_2^k, \dots, p_N^k\right\}, \quad \text{where } p_i^0 = x_i^0,\ \forall i. \tag{3}$$

We can say a particle in P as good or bad depending on its personal best being a good or bad point in P. Consequently, we call the ith particle (jth particle) in P the worst (the best) if $p_i^k(p_j^k)$ is the least (best) fitted, with respect to function value in P. The *pbest* and *gbest* is denoted as p_i^k and p_g^k, respectively.

At each iteration k, the position x_i^k of the ith particle is updated by a velocity v_i^{k+1} which depends on three components: its current velocity v_i^k, the cognition term (i.e., the weighted difference vectors $(p_i^k - x_i^k)$), and the social term (i.e., the weighted difference vector $(p_g^k - x_i^k)$).

Specifically, the set P is updated for the next iteration using

$$x_i^{k+1} = x_i^k + v_i^{k+1}, \tag{4}$$

where $v_i^{k+1} = v_i^k + r_1 \cdot c_1 \cdot (p_i^k - x_i^k) + r_2 \cdot c_2 \cdot (p_g^k - x_i^k)$.

The parameters r_1 and r_2 are uniformly distributed in random numbers in $[0, 1]$ and c_1 and c_2, known as the cognitive and social parameters, respectively, and are popularly chosen to be $c_1 = c_2 = 2.0$ [40]. Thus, the values $r_1 \cdot c_1$ and $r_2 \cdot c_2$ introduce some stochastic weighting in the difference vectors $(p_i^k - x_i^k)$ and $(p_g^k - x_i^k)$, respectively. The set P is updated as the new positions x_i^{k+1} that are created using the following rules with a minimization of the cost function:

$$p_i^{k+1} = x_i^{k+1} \quad \text{if } f\left(x_i^{k+1}\right) < f\left(p_i^k\right), \text{ otherwise } p_i^{k+1} = p_i^k. \tag{5}$$

This process of updating the velocities v_i^k, positions x_i^k, *pbest* p_i^k, and the *gbest* p_g^k is repeated until a user-defined stopping condition is met.

We now briefly present a number of improved versions of PSO and then show where our modified PSO can stand.

Shi and Eberhart [39] have done the first modification by introducing a constant inertia ω, which controls how much a particle tends to follow its current directions compared to the memorized *pbest* p_i^k and the *gbest* p_g^k. Hence, the velocity update is given by

$$v_i^{k+1} = \omega \cdot v_i^k + r_1 \cdot c_1 \cdot \left(p_i^k - x_i^k\right) + r_2 \cdot c_2 \cdot \left(p_g^k - x_i^k\right), \tag{6}$$

where the values of r_1 and r_2 are realized component wise.

Again Shi and Eberhart [40] proposed a linearly varying inertia weight during the search. the inertia weight is linearly reduced during the search. This entails a more globally search during the initial stages and a more locally search during the final stages. They also proposed a limitation of each particle's velocity to a specified maximum velocity $v^{\max}$. The maximum velocity was calculated as a fraction $\tau (0 < \tau \le 1)$ of the distance between the bounds of the search space, that is, $v^{\max} = \tau \cdot (x^u - x^l)$.

Fourie and Groenwold [41] suggested a dynamic inertia weight and maximum velocity reduction. In this modification, an inertia weight and maximum velocity are then reduced by fractions α and β, respectively, if no improvement in p_g^k occur after a prespecified number of iterations h, that is,

$$\text{if } f\left(p_g^k\right) = f\left(p_g^{k-1}\right) \quad \text{then } w_{k+1} = \alpha \omega_k \text{ and } v_k^{\max} = \beta v_k^{\max}, \tag{7}$$

where α and β are such that $0 < \alpha,\ \beta < 1$.

Clerc and Kennedy [42] introduced another interesting modification to PSO in the form of a constriction coefficient χ, which controls all the three components in velocity update rule. This has an effect of reducing the velocity as the search progress. In this modification, the velocity update is given by

$$\begin{gathered} v_i^{k+1} = \chi\left(v_i^k + r_1 c_1 \left(p_i^k\right) + r_2 c_2 \left(p_g^k - x_i^k\right)\right), \\ \text{where } \chi = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}, \qquad \phi = c_1 + c_2 > 4. \end{gathered} \tag{8}$$

Da and Ge [18] also modified PSO by introducing a temperature like control parameter as in the simulated annealing algorithm. Zhang et al. [43] have modified the PSO by introducing a new inertia weight during the velocity update. Generally in the beginning stages of their algorithm, the inertial weight ω should be reduced rapidly, when around optimum, the inertial weight ω should be reduced slowly. They adopted the following rule:

$$\begin{gathered} \omega = \omega_0 - \left(\frac{\omega_1}{\text{MAXITER1}}\right) * t, \quad \text{if } 1 \le t \le \text{MAXITER1}, \\ \omega = (\omega_0 - \omega_1) * \exp\left(\frac{(\text{MAXITER1} - k)}{\nu}\right), \\ \text{if MAXITER1} < k \le \text{MAXITER}, \end{gathered} \tag{9}$$

where ω_0 is the initial inertia weight, ω_1 is the inertial weight of linear section ending, MAXITER is the total searching generations, MAXITER1 is the used generations that inertia weight is reduced linearly, and k is a variable whose range is $[1, \text{MAXITER}]$. By adjusting k, they are getting different ending values of inertial weight.

In this work, the inertial weight is evolved as a part of searching the optimal sets of weights. However, the evolution of inertial weight is restricted between an upper limit (ω^u) and lower limit ω^l. If it exceeds the boundary during the course of training the network, then the following rule is adopted for restricting the value of ω:

$$\omega = \omega^l + \frac{c\_\text{value}}{3\omega^u}\left(\omega^u - \omega^l\right), \tag{10}$$

where $c_$value is the exceeded value.

In addition, the proposed method also uses the adaptive cognitive acceleration coefficient (c_1) and the social acceleration coefficients (c_2). c_1 has been allowed to decrease from its initial value of c_{1i} to c_{1f} while c_2 has been increased from c_{2i} to c_{2f} using the following equations as in [44]:

$$\begin{aligned} c_1^k &= \left(c_{1f} - c_{1i}\right)\frac{k}{\text{MAXITER}} + c_{1i}, \\ c_2^k &= \left(c_{2f} - c_{2i}\right)\frac{k}{\text{MAXITER}} + c_{2i}. \end{aligned} \tag{11}$$

3.3. ePSO-BP Learning Algorithm. The ePSO-BP is an learning algorithm which combines the ePSO global searching capability with the BP algorithm local searching capability. Similar to the GA [9], the ePSO algorithm is a global algorithm, which has a strong ability to find global optimistic result, and this ePSO algorithm, however, has a disadvantage that the search around global optimum is very slow. The BP algorithm, on the contrary, has a strong ability to find local optimistic result, but its ability to find the global optimistic result is weak. By combining the ePSO with the BP, a new algorithm referred to as ePSO BP hybrid learning algorithm is formulated in this paper. The fundamental idea for this hybrid algorithm is that at the beginning stage of searching for the optimum, the PSO is employed to accelerate the training speed. When the fitness function value has not changed for some generations, or value changed is smaller than a predefined number, the searching process is switched to gradient descending searching according to this heuristic knowledge. Similar to the ePSO algorithm, the ePSO BP algorithm's searching process is also started from initializing a group of random particles. First, all the particles are updated according to (4), until a new generation set of particles are generated, and then those new particles are used to search the global best (*gbest*) position in the solution space. Finally, the BP algorithm is used to search around the global optimum. In this way, this hybrid algorithm may find an optimum more quickly. The procedure for this ePSO BP algorithm can be summarized by the following computational steps.

(1) Initialize the positions and velocities of a group of particles randomly in the range of [0, 1]. Initialize the cognitive and social acceleration initial and final coefficients (i.e., c_{1i}, c_{1f}, c_{2i}, and c_{2f}).

(2) Evaluate each initialized particle's fitness value, and p_i is set as the positions of the current particles, while p_g is set as the best position of the initialized particles.

(3) If the maximal iterative generations are arrived, go to Step 10, else, go to Step 4.

(4) The best particle of the current particles is stored. The positions and velocities of all the particles are updated according to (4) and (6), then a group of new particles are generated.

(5) Adjust the value of c_1 and c_2 by using (11).

(6) Adjust the inertia weights ω according to equation (10) if it flies beyond the boundary of ω.

(7) Evaluate each new particle's fitness value, and the worst particle is replaced with the stored best particle. If the ith particle's new position is better than p_i, p_i is set as the new position of the ith particle. If the best position of all new particles is better than p_g, then p_g is updated.

(8) If the current p_g is unchanged for 15 consecutive generations, then go to Step 9; else, go to Step 3.

(9) Use the BP algorithm to search around p_g for some epochs, if the search result is better than p_g, output the current search result, or else, output p_g.

(10) Output the global optimum p_g.

The parameter ω, in the above ePSO BP algorithm, evolves simultaneously with the weights of the CFLANN during the course of training. The parameter MAXITER1 is generally adjusted to an appropriate value by many repeated experiments, then an adaptive gradient descending method is used to search around the global optimum p_g. The BP algorithm based on gradient descending has parameter called learning rate which controls the convergence of the algorithm to an optimal local solution. In practical applications, users usually employed theoretical, empirical, or heuristic methods to set a good value for this learning rate. In this paper, we adopted the following strategy for learning rate:

$$\mu = k * \exp(-\nu * \text{epoch}), \tag{12}$$

where μ is learning rate, k and ν are constants, epoch is a variable that represents iterative times, through adjusting k and ν and we can control the reducing speed of learning rate.

3.4. ePSO-BP Learning Algorithm for CFLNN. Learning of a CFLNN may be considered as approximating or interpolating a continuous multivariate function $\phi(X)$ by an approximating function $\phi_W(X)$. In CFLNN architecture, a set of basis functions φ 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 provide 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 [21].

Let k training patterns be applied to the FLNN and can be denoted by $\langle X_i, Y_i\rangle$, $i = 1, 2, \ldots, k$ and let the weight matrix be W. At the ith instant $i = 1, 2, \ldots, k$, the D-dimensional input pattern and the CFLNN output are given by $X_i = \langle x_{i1}, x_{i2}, \ldots, x_{iD}\rangle$, $i = 1, 2, \ldots, k$, and $\widehat{Y}_i = [\widehat{y}_i]$, respectively. Its corresponding target pattern is represented by $Y_i = [y_i]$, $i = 1, 2, \ldots, k$. Hence $\forall i, X = [X_1, X_2, \ldots, X_k]^T$. The augmented matrix of D-dimensional input pattern and the CFLNN output are given by

$$\left\langle X : \hat{Y} \right\rangle = \begin{pmatrix} x_{11} & x_{12} & . & x_{1D} & : & \widehat{y_1} \\ x_{21} & x_{22} & . & x_{2D} & : & \widehat{y_2} \\ . & . & . & . & : & . \\ . & . & . & . & : & . \\ x_{k1} & x_{k2} & . & x_{kD} & : & \widehat{y_k} \end{pmatrix}. \tag{13}$$

As the dimension of the input pattern is increased from D to D' by a set of basis functions φ, given by $\varphi(X_i) = [Ch_1(x_{i1}), Ch_2(x_{i1}),\ldots, Ch_1(x_{i2}), Ch_2(x_{i2})\ldots, Ch_1(x_{iD}), Ch_2(x_{iD}),\ldots]$. The $k \times D'$ 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 $W_i = [w_{i1}, w_{i2}, w_{i3},\ldots, w_{iD'}]$. The ith output of the CFLNN is given by $\hat{y}_i(t) = \rho(\Sigma_{j=1}^{D'}\psi_j(x_{ij}) \cdot w_{ij})\forall i$. The error associated with the ith output is given by $e_i(t) = y_i(t) - \hat{y}_i(t)$. Using the ePSO back-propagation (BP) learning, the weights of the CFLNN can be optimized. The high-level algorithms then can be summarized as follows.

(1) Input the set of given k training patterns.

(2) Choose the set of orthonormal basis functions.

(3) For $i = 1 : k$

(4) Expand the feature values using the chosen basis functions.

(5) Calculated the weighted sum and then fed to the output node.

(6) error = error + $e(k)$

(7) End for

(8) If the error is tolerable then stop otherwise go to (9).

(9) Update the weights using ePSO BP learning rules and go to step (3).

4. Empirical Study

This section is divided into five subsections. Section 4.1 describes the datasets taken from UCI [45] repository of machine learning databases. The parameters required for the proposed method are given in Section 4.2. The performance of the hybrid CFLNN using some of the datasets especially considered by Sierra et al. [21] compared with the model proposed by Sierra et al. in Section 4.3. In Section 4.4, the classification accuracy of hybrid CFLNN is compared with FLNN [8]. In Section 4.5, we compared the performance of hybrid CFLNN with FLNN proposed in [8] using the cost matrix analysis and then compared with the results obtained by StatLog project [46].

4.1. Description of the Datasets. The availability of results, with previous evolutionary and constructive algorithms (e.g., Sierra et al. [21], Preshelt [47]) has guided us the selection of the following varied datasets taken from the UCI repository of machine learning databases for the addressed neural network learning. Table 1 presents a summary of the main features of each database that has been used in this study.

Table 1: Summary of the datasets.

Dataset	Patterns	Attrib.	Clas.	Patterns in class 1	Patterns in class 2	Patterns in class 3
IRIS	150	4	3	50	50	50
WINE	178	13	3	71	59	48
PIMA	768	8	2	500	268	—
BUPA	345	6	2	145	200	—
HEART	270	13	2	150	120	—
CANCER	699	9	2	458	241	—

Table 2: Description of the parameters.

Symbol	Purpose of the symbol
N	Size of the swarm
ω	Inertia weight
ω^u	Upper limit of the inertia
ω^l	Lower limit of the inertia
c_1	Cognitive parameter
c_{1i}	Left boundary value of cognitive parameter
c_{1f}	Right boundary value of cognitive parameter
c_2	Social parameter
c_{2i}	Left boundary value of social parameter
c_{2f}	Right boundary value of social parameter
MAXITER	Maximum iterations for stopping an algorithm

4.2. Parameters. All the algorithms have some parameters that have to be provided by the user. The parameters for the proposed hybrid CFLNN are listed in Table 2. However, the parameters for other algorithms are set based on the suggestion. The parameters for EFLN were adopted as suggested in [21]. Similarly, the parameters for FLNN were set as suggested in [8].

The values of the parameters used in this paper are as follows. We set $N = 20 * d$, where d is the dimension of the problem under consideration. The upper limit (ω^u) and lower limit (ω^l) of the inertia are set to $[0.2, 1.8]$. Similarly, the initial and final value of cognitive acceleration coefficients are set to $c_{1i} = 2.5$ and $c_{1f} = 0.5$. The initial and final value of social acceleration coefficients are set to $c_{2i} = 0.5$ and $c_{2f} = 2.5$. the maximum number of iteration is fixed to MAXITER = 500.

In the case of BP learning, the learning parameter μ and the momentum factor ν in hybrid CFLNN was chosen after a several runs to obtain the best results. In the similar manner, the functional expansion of the hybrid CFLNN was carried out.

4.3. Hybrid CFLNN versus EFLN. In this subsection, we will compare the results of hybrid CFLNN with the results of EFLN with polynomial basis functions of degree 1, 2, and 3. The choice of the polynomial degree is obviously a key question in FLNN with polynomial basis functions. However, Sierra et al. [21] have given some guidance to

Table 3: Possible number of expanded inputs of degrees ONE, TWO, and THREE.

Dataset	Attributes	Degree 1	Degree 2	Degree 3
IRIS	4	5	15	35
WINE	13	14	105	560
PIMA	8	9	45	165
BUPA	6	7	28	84
HEART	13	14	105	560
CANCER	9	10	55	220

Table 4: Comparative results of HCFLNN with EFLN for the cancer and PIMA dataset by considering the average training error (MTre), average validation error (MVe), and average test error (MTe).

Dataset	HCFLNN			EFLN		
	MTre	MVe	MTe	MTre	MVe	MTe
Cancer 1	4.01	2.76	2.57	4.27	1.89	2.09
Cancer 2	3.95	3.97	4.66	4.37	2.96	3.96
cancer 3	4.13	3.51	4.43	3.29	3.01	4.65
BUPA 1	16.26	21.98	22.62	19.07	22.44	23.29
BUPA 2	17.90	24.12	22.35	19.84	18.63	20.37
BUPA 3	15.34	19.92	21.96	16.68	17.81	24.44

optimize the polynomial degree that can best suit to the architecture. Considering degrees of the polynomial 1, 2, and 3, the possible number of expanded inputs of the above datasets are given in Table 3.

For the sake of convenience, we report the results of the experiments conducted on CANCER and BUPA and then compared with the methods EFLN [21]. We partitioned both datasets into three sets: training, validation, and test sets. Both the networks are trained for 1500 epochs (it should be carefully examined) on the training set, and the error on the validation set was measured after every 10 epochs. Training was stopped when a maximum of 1500 epochs had been trained. The test set performance was then computed for that state of the network which had minimum validation set error during the training process. This method called early stopping is a good way to avoid overfitting of the network to the particular training examples used, which would reduce the generalization performance. The average error rate corresponding to HCFLNN, and EFLN w.r.t. training, validation, and testing of CANCER, and BUPA datasets are shown in Table 4.

4.4. Hybrid CFLNN versus FLNN. Here, we will discuss the comparative performance of hybrid CFLNN with FLNN using three datasets IRIS, WINE, and PIMA. In this case, the total set of samples are randomly divided into two equal folds. Each of these two folds are alternatively used either as a training set or as a test set. As the proposed learning method ePSO BP learning is a stochastic algorithm, so 10 independent runs were performed for every single fold. The training results obtained in the case of HCFLNN, averaged over 10 runs, are compared with the single run of FLNN. Similarly, the performance of both classifiers in test set is illustrated herein.

Table 5: Comparative average performance of HCFLNN and FLNN [21] based on the confidence level ($\alpha = 95\%$).

Dataset	HCFLNN	FLNN
IRIS train	0.9964 ± 0.0136	0.9866 ± 0.0260
Test set	0.9864 ± 0.0262	0.9866 ± 0.0260
WINE train	0.9842 ± 0.0259	0.9605 ± 0.0405
Test set	0.9708 ± 0.0350	0.9550 ± 0.0431
PIMA train	0.8064 ± 0.0395	0.7877 ± 0.0409
Test set	0.7928 ± 0.0405	0.7812 ± 0.0414

Table 6: Comparative Average Performance of HCFLNN and FLNN [8] based on the Confidence Level ($\alpha = 98\%$).

Dataset	HCFLNN	FLNN
IRIS Train	0.9964 ± 0.0161	0.9866 ± 0.0309
Test set	0.9864 ± 0.0312	0.9866 ± 0.0309
WINE Train	0.9842 ± 0.0308	0.9605 ± 0.0481
Test set	0.9708 ± 0.0416	0.9550 ± 0.0512
PIMA Train	0.8064 ± 0.0470	0.7877 ± 0.0486
Test set	0.7928 ± 0.0482	0.7812 ± 0.0492

Table 7: Weight Matrix of classes to Penalize.

Real Classification	Model Classification	
	Class 1	Class 2
Class 1	0	ω_2
Class 2	ω_1	0

The plotted results clearly indicate that the performance of HCFLNN is competitive with FLNN, whereas in other classification problems like WINE and PIMA, the HCFLNN is showing a clear boundary.

The comparative performance of HCFLNN with FLNN [8] is given in Tables 5 and 6 w.r.t to the different confidence level (α) of 95% and 98%, respectively.

4.5. Performance of Hybrid CFLNN versus FLNN Based on Heart Data. In this subsection, we will explicitly examine the performance of the HCFLNN model by considering the heart dataset with the use of the 9-fold cross validation methodology. The reason for using 9-fold cross validation is that to compare the performance with the performance of few of the representative algorithms considered in StatLog Project [46]. In 9-fold cross validation, we partition the database into nine subsets (heart1.dat, heart2.dat,..., heart9.dat), where eight subsets are used for training, and the remaining one is used for testing. The process is repeated nine times in such a way that each time a different subset of data is used for testing. Thus, the dataset was randomly segmented into nine subsets with 30 elements each. Each subset contains about 56% of samples from class 1 (without heart disease) and 44% of samples from class 2 (with heart disease).

The procedure makes use of a weight matrix, which is described in Table 7.

Table 8: Heart disease classification performance of FLANN models.

Data subset	Error in training set		Error in test set		C_{train}	C_{test}
	Class 1	Class 2	Class 1	Class 2		
Heart1	13/133	14/107	1/17	1/13	0.35	0.2
Heart2	14/133	12/107	2/17	1/13	0.31	0.23
Heart3	13/134	15/106	4/16	2/14	0.37	0.47
Heart4	13/133	10/107	1/17	4/13	0.26	0.7
Heart5	13/133	16/107	3/17	2/13	0.39	0.43
Heart6	13/134	14/106	6/16	0/14	0.35	0.2
Heart7	15/133	13/107	0/17	3/13	0.33	0.5
Heart8	18/133	17/107	1/17	0/13	0.43	0.03
Heart9	20/134	9/106	2/16	1/14	0.27	0.23
Mean					0.34	0.33

Table 9: Heart disease classification performance of HCFLANN models.

Data subset	Error in training set		Error in test set		C_{train}	C_{test}
	Class 1	Class 2	Class 1	Class 2		
Heart1	13/133	14/107	1/17	1/13	0.35	0.2
Heart2	13/133	12/107	1/17	2/13	0.30	0.36
Heart3	12/134	13/106	5/16	1/14	0.32	0.33
Heart4	13/133	10/107	4/17	1/13	0.26	0.30
Heart5	13/133	15/107	3/17	2/13	0.37	0.43
Heart6	13/134	12/106	5/16	1/14	0.30	0.30
Heart7	14/133	13/107	1/17	2/13	0.33	0.37
Heart8	16/133	16/107	0/17	2/13	0.40	0.33
Heart9	18/134	10/106	2/16	1/14	0.28	0.23
Mean					0.32	0.31

The purpose of such a matrix is to penalize wrongly classified samples based on the weight of the penalty of the class. In general, the weight of the penalty for class 2 samples that are classified as class 1 samples is ω_1, while the weight of the penalty for class 1 records that are classified as class 2 samples is ω_2. Therefore, the metric used for measuring the cost of the wrongly classifying patterns in the training and test dataset is given by (14).

$$C_{\text{train}} = \frac{(S_1 \times \omega_1 + S_2 \times \omega_2)}{S_{\text{train}}},$$
$$C_{\text{test}} = \frac{(S_1 \times \omega_1 + S_2 \times \omega_2)}{S_{\text{test}}}, \tag{14}$$

where C_{train} is the cost of the training set; C_{test} is the cost of test set; S_1 and S_2 denote the patterns that are wrongly classified as belong to class 1 and 2, respectively; S_{train} and S_{test} are the total number of training and test patterns, respectively.

Table 8 presents the errors and costs of the training and test sets for the FLANN model with a weight value of $\omega_1 = 5$ and $\omega_2 = 1$.

Table 9 illustrates the performance of HCFLANN based on the above definition of cost matrix. The errors in training and test set are explicitly given.

The classification results found by the HCFLNN for the heart disease dataset were compared with the results found in the StatLog project [46]. According to [46], comparison consists of calculating the average cost produced by the nine data subsets used for validation. Table 10 presents the average cost for the nine training and test subsets. The result of the HCFLNN is highlighted in bold.

5. Conclusions and Research Directions

In this paper, we developed a new hybrid Chebyshev functional link neural network (HCFLNN). The hybrid model is constructed using the newly proposed ePSO- back propagation learning algorithm and functional link artificial neural network with the orthogonal Chebyshev polynomials. The model was designed for the task of classification in data mining. The method was experimentally tested on various benchmark datasets obtained from publicly available UCI repository. The performance of the proposed method demonstrated that the classification task is quite well in WINE and PIMA whereas showing a competitive performance with FLNN in IRIS. Further, we compared this model with EFLN and FLNN, respectively. The comparative results of the developed model is showing a clear edge over FLNN. Compared with EFLN, the proposed method has been shown to yield state-of-the-art recognition error rate for the classification problems such as CANCER and BUPA.

Table 10: Comparative classification performance of HCFLNN, FLNN with the algorithms considered in [46] using the heart disease bench mark datset.

Methods	C_{test}	C_{train}
HCFLNN	0.31	0.32
FLNN	0.33	0.34
HNFB^{-1}	0.37	0.59
Bayes	0.37	0.35

With this encouraging results of HCFLNN, our future research includes: (i) testing the proposed method on a more number of real life bench mark classification problems with highly nonlinearly boundaries, (ii) mapping the input features with other polynomials such as Legendre, Gaussian, Sigmoid, power series, and so forth, for better approximation of the decision boundaries, (iii) the stability and convergence analysis of the proposed method, and (iv) the evolution of optimal FLNN using particle swarm optimization.

The HCFLNN architecture, because of its simple architecture and computational efficiency, may be conveniently employed in other tasks of data mining and knowledge discovery in databases [4, 8] such as clustering, feature selection, feature extraction, association rule mining, regression, and so on. The extra calculation generated by the higher-order units can be eliminated, provided that these polynomial terms are stored in memory instead of being recalculated each time the HCFLNN trained.

References

[1] J. Ghosh and Y. Shin, "Efficient higher-order neural networks for classification and function approximation," *International Journal of Neural Systems*, vol. 3, pp. 323–350, 1992.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Englewood Cliffs, NJ, USA, 1999.

[3] O. L. Mangasarian and E. W. Wild, "Nonlinear knowledge-based classification," *IEEE Transactions on Neural Networks*, vol. 19, no. 10, pp. 1826–1832, 2008.

[4] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[5] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order neural networks," *Applied Optics*, vol. 26, no. 23, pp. 4972–4978, 1987.

[6] Y. H. Pao, *Adaptive Pattern Recognition and Neural Network*, Addison-Wesley, Reading, Mass, USA, 1989.

[7] E. Artyomov and O. Yadid-Pecht, "Modified high-order neural network for invariant pattern recognition," *Pattern Recognition Letters*, vol. 26, no. 6, pp. 843–851, 2005.

[8] B. B. Misra and S. Dehuri, "Functional link neural network for classification task in data mining," *Journal of Computer Science*, vol. 3, no. 12, pp. 948–955, 2007.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Morgan Kaufmann, 1989.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Pisacataway, NJ, USA, December 1995.

[11] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: a survey of the state of the art," in *Proceedings of International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pp. 1–37, 1992.

[12] Y. Davidor, "Epistasis variance: suitability of a representation to genetic algorithms," *Complex Systems*, vol. 4, pp. 368–383, 1990.

[13] L. J. Eshelman and J. D. Schaffer, "Real coded genetic algorithms and interval schemata," in *Foundation of Genetic Algorithms*, L. D. Whitley, Ed., pp. 187–202, Morgan Kaufmann, 1993.

[14] H. Muhlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm I. Continuous parameters optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 24–49, 1993.

[15] J. F. Schutte and A. A. Groenwold, "A study of global optimization using particle swarms," *Journal of Global Optimization*, vol. 31, no. 1, pp. 93–108, 2005.

[16] M. M. Ali and P. Kaelo, "Improved particle swarm algorithms for global optimization," *Applied Mathematics and Computation*, vol. 196, no. 2, pp. 578–593, 2008.

[17] J. Yu, S. Wang, and L. Xi, "Evolving artificial neural networks using an improved PSO and DPSO," *Neurocomputing*, vol. 71, no. 4–6, pp. 1054–1060, 2008.

[18] Y. Da and X. R. Ge, "An improved PSO-based ANN with simulated annealing technique," *Neurocomputing*, vol. 63, pp. 527–533, 2005.

[19] M. S. Klassen, Y. H. Pao, and V. Chen, "Characteristics of the functional link net: a higher order delta rule net," in *Proceedings of the 2nd Annual International Conference on Neural Networks*, vol. 1, pp. 507–513, San Diago, Calif, USA, 1988.

[20] Y. H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.

[21] A. Sierra, J. A. Macías, and F. Corbacho, "Evolution of functional link networks," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 54–65, 2001.

[22] B. Haring and J. N. Kok, "Finding functional links for neural networks by evolutionary computation," in *Proceedings of the 5th Belgian-Dutch Conference on Machine Learning (BENELEARN '95)*, T. Van de Merckt et al., Ed., pp. 71–78, Brussels, Belgium, 1995.

[23] J. C. Patra and R. N. Pal, "A functional link artificial neural network for adaptive channel equalization," *Signal Processing*, vol. 43, no. 2, pp. 181–195, 1995.

[24] S. Haring, J. N. Kok, and M. C. van Wezel, "Feature selection for neural networks through functional links found by evolutionary computation," in *Proceedings of the 2nd International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data*, X. Liu et al., Ed., vol. 1280 of *Lecture Notes in Computer Science*, pp. 199–210, 1997.

[25] 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*, vol. 7, no. 3, pp. 209–221, 1999.

[26] D. A. Panagiotopoulos, R. W. Newcomb, and S. K. Singh, "Planning with a functional neural-network architecture," *IEEE Transactions on Neural Networks*, vol. 10, no. 1, pp. 115–127, 1999.

[27] 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*, vol. 29, no. 2, pp. 254–262, 1999.

[28] J. C. Patra and A. van den Bos, "Modeling of an intelligent pressure sensor using functional link artificial neural networks," *ISA Transactions*, vol. 39, no. 1, pp. 15–27, 2000.

[29] 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*, vol. 32, no. 4, pp. 505–511, 2002.

[30] S. N. Singh and K. N. Srivastava, "Degree of insecurity estimation in a power system using functional link neural network," *European Transactions on Electrical Power*, vol. 12, no. 5, pp. 353–358, 2002.

[31] T. Marcu and B. Koppen-Seliger, "Dynamic functional link neural networks genetically evolved applied to system identification," in *Proceedings of European Symposium on Artificial Neural Networks (ESANN '04)*, pp. 115–120, Bruges, Belgium, 2004.

[32] B. Majhi and H. Shalabi, "An improved scheme for digital watermarking using functional link artificial neural network," *Journal of Computer Science*, vol. 1, no. 2, pp. 169–174, 2005.

[33] 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*, vol. 34, no. 3, pp. 261–277, 2005.

[34] S. Purwar, I. N. Kar, and A. N. Jha, "On-line system identification of complex systems using Chebyshev neural networks," *Applied Soft Computing Journal*, vol. 7, no. 1, pp. 364–372, 2007.

[35] W. D. Weng, C. S. Yang, and R. C. Lin, "A channel equalizer using reduced decision feedback Chebyshev functional link artificial neural networks," *Information Sciences*, vol. 177, no. 13, pp. 2642–2654, 2007.

[36] D. Krishnaiah, D. M. 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*, vol. 3, no. 4, pp. 90–96, 2008.

[37] J. C. Patra, G. Chakraborty, and S. Mukhopadhyay, "Functional link neural network-based intelligent sensors for harsh environments," *Sensors & Transducers Journal*, vol. 90, pp. 209–220, 2008.

[38] S. Dehuri, B. B. Mishra, and S.-B. Cho, "Genetic feature selection for optimal functional link artificial neural network in classification," in *Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL '08)*, C. Fyfe et al., Ed., vol. 5326 of *Lecture Notes in Computer Science*, pp. 156–163, 2008.

[39] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, IEEE Press, Pisacataway, NJ, USA, May 1998.

[40] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 591–600, Springer, Berlin, Germany, 1998.

[41] P. C. Fourie and A. A. Groenwold, "The particle swarm optimization algorithm in size and shape optimization," *Structural and Multidisciplinary Optimization*, vol. 23, no. 4, pp. 259–267, 2002.

[42] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[43] J. R. Zhang, J. Zhang, T. M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1026–1037, 2007.

[44] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[45] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," http://www.ics.uci.edu/mlearn/MLRepository.html.

[46] R. P. Lippmann, "An introduction to computing with neural networks," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[47] L. Preshelt, "Proben1-a set of neural network benchmark problems and benchmarking rules," Tech. Rep. 21/94, Universitat Karlsruhe, Karlsruhe, Germany, 1994.