Neural Comput & Applic
DOI 10.1007/s00521-009-0288-5

KES 2008

A comprehensive survey on functional link neural networks and an adaptive PSO–BP learning for CFLNN

Satchidananda Dehuri · Sung-Bae Cho

Received: 14 March 2009 / Accepted: 14 June 2009

Abstract Functional link neural network (FLNN) is a class of higher order neural networks (HONs) and have gained extensive popularity in recent years. FLNN have been successfully used in many applications such as system identification, channel equalization, short-term electric-load forecasting, and some of the tasks of data mining. The goals of this paper are to: (1) provide readers who are novice to this area with a basis of understanding FLNN and a comprehensive survey, while offering specialists an updated picture of the depth and breadth of the theory and applications; (2) present a new hybrid learning scheme for Chebyshev functional link neural network (CFLNN); and (3) suggest possible remedies and guidelines for practical applications in data mining. We then validate the proposed learning scheme for CFLNN in classification by an extensive simulation study. Comprehensive performance comparisons with a number of existing methods are presented.

Keywords Classification · Functional link neural networks · Chebyshev functional link neural network · Particle swarm optimization · Back-propagation learning

S. Dehuri (✉)
Department of Information and Communication Technology,
Fakir Mohan University, Vyasa Vihar,
Balasore 756019, Orissa, India
e-mail: satchi.lapa@gmail.com

S.-B. Cho
Soft Computing Laboratory, Department of Computer Science,
Yonsei University, 262 Seongsanno, Seodaemun-gu,
Seoul 120-749, Korea
e-mail: sbcho@cs.yonsei.ac.kr

1 Introduction

In recent years, artificial neural networks (ANNs) [1–5] have gained extensive popularity. Research activities from McCulloch–Pitts model of neuron [6] to higher order neuron [7] are considerable and the literature is growing day-by-day. ANN have been used increasingly as a promising modeling tool in almost all areas of human activities where quantitative approaches can be used to help decision making (e.g., functional approximation [8–10], rule extraction [11–14], pattern recognition and classification [15–17], forecasting and prediction [18–31], business [32, 33], civil engineering [34], electrical engineering [35], and medical area [36–38]). Indeed ANNs have already been treated as a standard nonlinear alternative to traditional models for pattern classification, time series analysis, and regression problems. ANNs are capable of generating complex mapping between input and output space, therefore, arbitrarily complex nonlinear decision boundaries can be approximated by these networks.

In spite of the development of various types of ANNs, higher order neural networks (HONs) [39–41] have recently been treated as an attractive method to eradicate some of the limitations that exist in non-HONs. Although the main focus of this paper is on FLNN, it is also very crucial to introduce the HONs because the root of FLNN is HON. The following few lines can guide the readers about the motivation behind HONs.

Feed-forward networks based on single layer of linear threshold logic units exhibit fast learning, e.g., the simple perceptron [42] can realize a linearly separable dichotomy in a finite number of learning steps, however, does not converge or approximate non-linear decision boundaries. Adaline [43, 44] tries to find a solution which minimizes a mean square error criterion by using gradient descent

Published online: 01 July 2009

learning algorithm to adjust the weights. However, it discrimination capability is also limited to linearly separable problems as in the case of the single-layered perceptron. The fraction of dichotomies that are linearly separable drastically reduces with increase in input dimension [45]. Moreover, many real problems involve approximating non-linear functions or forming multiple non-linear decision regions. This limits the applicability of simple single-layered networks of linear threshold units.

The addition of a layer of hidden units dramatically increases the power of layered feed-forward networks. Indeed, networks with a single hidden layer using arbitrary squashing functions are capable of approximating any measurable function from one finite dimensional space to another to any desired degree of accuracy provided sufficiently many hidden units are available [46]. In particular, the multi-layer perceptron (MLP) using the back-propagation (BP) learning algorithm has been successfully applied to many applications. However, the training speeds for MLP are typically much slower than those for feed-forward networks comprising a single layer of linear threshold units due to BP of error induced by multilayering. Moreover, the problems such as local minima trapping, saturation, weight interference, initial weight dependence, and overfitting, make MLP training difficult. Additionally, it is also very difficult to fix the parameters like number of neurons in a layer, and number of hidden layers in a network, thereby deciding a proper architecture is not that easy.

An easy way to avoid these problems consists of 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 [47, 48]. In other words, higher-order correlations among input components can be used to construct a higher-order network to perform non-linear mappings using only a single layer of units [7]. All these prompted the field of HONs [49, 50] like functional link neural networks (FLNNs) [51–54], ridge polynomial neural networks (RPNNs) [55–57], pi-sigma neural networks [58–65], radial basis functions [66–70] and so on.

Since then several authors, including Lee et al. [71], Peretto and Niez [72], Psaltis and Park [73], Gardner [74], Abbott and Arian [75], and Horn and Usher [77] (see also [76], Chap. 5]), have demonstrated the improved capacity of higher-order neural networks by simulations and by lower bounds on the capacity that are derived using a nonrigorous analysis.

Thus far we have discussed where is the root of FLNN but not yet discussed the detail trends of FLNN. In Sect. 3 we will give a complete road map of FLNN. Let us discuss the motivation behind the second achievement of this paper.

In the context of training FLNN, the mostly used algorithm is the BP-learning algorithms (FLNN with various type of learning schemes are given in Tables 1 and 2 of Sect. 3). Hence, the inherent problems that exist in BP-learning algorithm [78–80] are also frequently encountered in the use of this algorithm. First, the

Table 1 Summary of the FLNNs (1)

Reference	Method	Application	Basis Function	Learning
[81]	ClFLNN	Classification	Polynomial	Pseudoinverse
[82]	PFLNN	Planning	Polynomial	–
[83]	SiFLNN	System identification	Trigonometric	BP
[84]	EFLN	Classification	Polynomial	GA + BP
[85]	SyFLNN	System identification	Trigonometric	DBP
[86]	CeFLNN	Channel equalization	Trigonometric	BP
[87]	COFLNN	Channel equalization	Chebyshev	NLMS
[88]	ClaFLNN	Classification	Trigonometric	BP
[89]	IpFLNN	Intelligent pressure sensor	Chebyshev, Legendre, power series	BP
[90]	GFLNN	Classification	Trigonometric	BP
[51]	FLANN	Classification	Trigonometric	BP
[91]	DwFLNN	Digital watermarking	Trigonometric	BP
[92]	IsFLNN	Intelligent sensors	Trigonometric	BP
[93]	ElfFLNN	Electric load forecasting	Trigonometric	BP
[94]	CcFLNN	Carrageenan concentration	Polynomial	EBP
[95]	IeFLNN	Insecurity estimation	Trigonometric	Adaptive
[96]	ClasFLNN	Classification	Sigmoidal	Adaptive

BP back propagation, *EBP* error back propagation, *DBP* dynamic back propagation, *NLMS* normalized least mean square

Table 2 Summary of the FLNNs (2)

Reference	Method	Application	Basis Function	Learning
[97]	BpFLNN	Bankruptcy prediction	Polynomial	Genetic algorithm
[98]	DBRVFLNN	Off-line recognition of English script	Random vector	BP
[99]	IraFLNN	Interval regression analysis	Trigonometric	Genetic algorithm
[100]	AFLNN	Function approximation	Random vector	BP
[54]	CoFLNN	Control	Random vector	Delta rule conjugate gradient
[101]	RDFFLNN	Channel equalization	Trigonometric	BP
[102]	DfFLNN	CCI in DCS	Trigonometric	MSE
[103]	QsFLNN	QAM Signal	Trigonometric	BP
[104]	SyiFLNN	System identification	Chebyshev	BP
[105]	RDFCFLNN	Channel equalization	Chebyshev	BP
[106]	CFLNN	System identification	Chebyshev	BP

BP back propagation, *MSE* mean squared error, *CCI* co-channel interference, *DCS* digital communication system

BP-learning algorithm easily get trapped in local optima especially for those non-linearly separable classification problems. Second, the convergence speed of the BP learning is too slow even if the learning goal, a given termination error, can be achieved. In addition, the convergence behavior of the BP-learning algorithm depends very much on the choices of initial values of the network connection weights as well as the parameters in the algorithm such as the learning rate and momentum. But 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.

We can harness the power of genetic algorithms (GAs) [107–109] and particle swarm optimization (PSO) [110] for training the FLNN to reduce the local optimality and speed up the convergence. But training using genetic algorithm is not advisable because of the following limitations: in the training process it requires encoding and decoding operators which are commonly treated as a long standing barrier of neural networks researchers. The problem of applying genetic algorithms to train neural networks may be unsatisfactory because recombination operators incur several problems, such as competing conventions [111] and the epistasis effect [112]. For better performance, real coded genetic algorithms [113, 114] 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 have some attractive properties; it retains previous useful information, whereas GA 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 [115, 116] and neural networks design [117, 118] have demonstrated its potential. It is considered to be capable of reducing 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 crossover 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 algorithms 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 slowly near the global optimum. Hence to cope with this problem, we suggested an adaptive PSO (aPSO) and back-propagation (BP) algorithm as a learning method of Chebyshev functional link neural network (CFLNN) for fine tuning the connection weights. The rationale of Chebyshev basis functions is given in Sect. 4.

The concept of Chebyshev functional link neural network is not new, but its usage for approximation of decision boundaries in classification problem is the art of this work. The training CFLNN uses the aPSO for global search in the beginning stage, and then BP-learning algorithm to do local search around the global optimum *gbest*. We have shown its effectiveness of classifying the unknown pattern using the publicly available datasets obtained from

University of California, Irvine (UCI) [119] repository. The computational results are then compared with FLNN [51–56] with a generic basis functions and EFLN. From the comparative study we observed that the performance of the proposed one outperforms FLNN and EFLN in terms of classification accuracy.

Findings A clear road map of FLNNs development over the years and their success in various fields. FLNN with a new learning scheme for classification.

Originality/value The reader will appreciate the comprehensive survey of FLNNs and the proposed learning scheme for CFLNN in classification, and hopefully be encouraged to explore further the possibility of using the proposed method to achieve improved performance in their own applications.

Outline The rest of this paper is organized as follows. The basic architecture of functional link neural networks (FLNNs) and various learning schemes are discussed in Sect. 2. Section 3 provides a road map of how FLNN has developed over time, which will then provide readers with the means to understand the proposed one. In Sect. 4, we have presented the proposed learning scheme for CFLNNs and its application in classification. Experimental results are demonstrated in Sect. 5. Section 6 concludes the article with future research areas of FLNN.

2 Architectures and learning of FLNN

In this section we will discuss the basic architecture and learning of FLNN. Although, there are a wide variety of architectures and learning schemes of FLNN so far developed, we identified four most predominant architectures and their learning schemes for sharpening the knowledge. These architectures are varied based on their usage of basis function for enhancement of input patterns.

2.1 Random variable functional link neural networks (RVFLN)

Pao et al. in 1992 [120] has proposed a functional link neural network named as random vector functional link network (RVFLN). The theoretical basis of RVFLN is primarily based on Theorem 2 by Hornik et al. [46, 121].

The following definition and theorems can be the primary theoretical realization of RVFLN.

Definition For any measurable function $\chi(.) : R^r \to R$ and $r \in N$, $\sum \prod^r(\chi)$ be the class of functions, $\{f : R^r \to R : \quad f(x) = \sum_{j=1}^{q} \beta_j \cdot \Pi_{k=1}^{l_j} \chi(A_{jk}(x)), x \in R^r, \beta_j \in R, A_{jk} \in A^r,\ l_j \in N, q = 1, 2, \ldots\}$, where A^r is the set of all affine functions from R^r to R. For the special case of $l_j = 1$, we have the $\sum$-networks.

Theorem 1 For every continuous nonconstant function χ, every r, and every probability measure μ on (R^r, B^r), $\sum \prod^r(\chi)$ is ρ_μ dense in M^r, where μ is a probability measure taken for convenience to describe the relative frequency of occurrence of input patterns μ, B^r is the Borel field of R^r, and M^r is the set of all Borel measurable functions from R^r to R.

According to Hornik et al. [46, 121], the significance of this theorem is that single hidden layer feed-forward networks can approximate any measurable function arbitrarily well, regardless of the continuous nonconstant function χ used, regardless of the dimension of the input space r, and the input space environment μ.

Theorem 2 For every squashing function Ψ, every r, and μ on (R^r, B^r), $\Sigma^r(\Psi)$ is uniformly dense on compacta in C^r and ρ_μ dense in M^r (where C^r is the set of continuous functions from R^r to R, and C^r is a subset of M^r).

This theorem says that if a function f is a mapping from R^r to R, then that function can be approximated arbitrarily well by

$$f = \sum_{j=1}^{q} \beta_j \chi(A_j x + b_j). \tag{1}$$

This can be implemented in terms of a single hidden layer feed forward net with all weights A_j and β_j to be learned (as well as the thresholds) with BP, or it can be implemented with a flat net known as functional link neural net.

The RVFLN generates A_j and b_j randomly, and must learn only β_j. This results in a flat-net architecture for which only weights β_j must be learned. Learning is the nature of quadratic optimization and is extremely rapid.

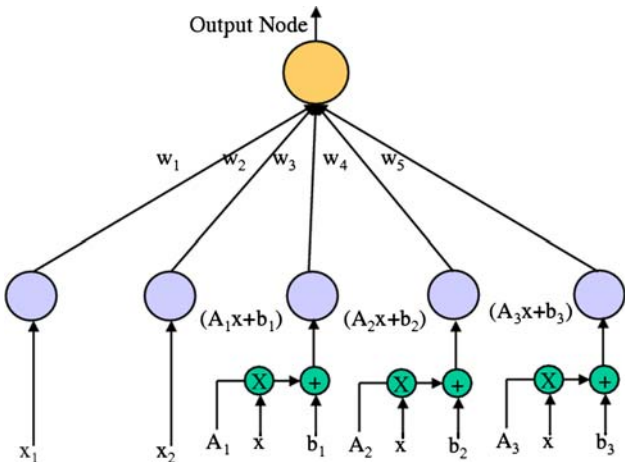

Fig. 1 Random vector functional link network

The system architecture is shown in Fig. 1. The simple algorithm for RVFLN can be described as follows: for supervised learning the inputs are enhanced to x_p with elements (x_{p1}, x_{p2}, …, x_{pj}) and let the target output of the pattern x_p be t_p. Initially, the algorithm assigns the weight β_j random values. It calculates the output o_p linearly as $o_p = \Sigma\beta_j x_{pj}$. For each input pattern the changes in the weights are taken to be

$$\Delta\beta_{pj} = \eta(t_p - o_p)x_{pj}. \tag{2}$$

The changes are calculated for all the patterns in the training set, and after each such presentation the weights are updated according to

$$\beta_j(k+1) = \beta_j(k) + \Sigma_p \Delta\beta_{pj}. \tag{3}$$

Updating is continued until the values of the weights β_j does not change significantly. The value of the parameter η may be increased as $t_p - o_p$ decreases.

2.2 Functional link neural networks with generic basis

In this architecture the input pattern of a functional-link net is a tensor representation. Pao [48] demonstrated that the tensor representation is very effective in classification. Therefore, it is important for us to discuss this model as an alternative to RVFLN. In the tensor model, a feature of x, say x_i, can be enhanced as $x_i x_j$, $x_i x_j x_k$, and $x_i x_j x_k x_l$ and so on, where $i \le j \le k \le l$. It is observed that an original tensor representation contains many higher-order terms. For instance, an enhanced pattern of $x = (x_1, x_2, x_3)$ with the tensor representation can be generated as ($x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3, x_1^2x_2, x_1^2x_3, x_1x_2^2, x_2^2x_3, x_1x_3^2, x_2x_3^2, x_1^3, x_2^3, x_3^3, x_1x_2x_3$). It can be seen that a large number of terms in the tensor representation will be generated as the dimensions of x increases. Hence, Pao [48] suggested that higher-order terms beyond the second order such as $x_2x_3^2$ and $x_1x_2x_3$, are not required. In addition, two or more equal indices in the enhanced pattern should be omitted. However, it needs proper investigation. For instance, an acceptable tensor representation of (x_1, x_2, x_3) is ($x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3$).

A functional-link net with single output node and $n + n(n-1)/2$ input nodes is a one-layer feed-forward network, where y is the actual output corresponding to the input pattern x, and n is the number of input features. Let the tensor of x be represented as $x_t = \langle x_1, x_2, \ldots, x_n, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n\rangle$. Let f denote the output node's activation function or sigmoid function, and θ be a bias in f. Without loss of generality, f is defined as follows:

$$f(s) = \frac{1}{1+\exp(-s)}, \tag{4}$$

where s is equal to $wx_t - \theta$ such that $\hat{y} = f(s)$. Here, wx_t is an aggregated value which is the inner product of w with x_t. In principle, x can be categorized as one class or the other class if the sigmoid function's output value is not below or below 0.5 [38]. Thus, square errors denoted by E between the actual and desired outputs of individual training patterns can be measured as

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{5}$$

where y_i and $\hat{y}_i$ are the desired and actual output of the ith input training pattern, respectively, and n is the number of training pattern. By the well-known BP-learning algorithm [122], the training phase is continued to update w and θ until a convergent condition is reached. For instance, if E is below a pre-specified value, then the learning algorithm can be stopped.

2.3 Functional link neural networks with trigonometric basis

The basic architecture of this model is same as the previous two models, but only difference we can observe here is in the selection of basis functions. The details of this model is discussed in [83] and some of the important points are analyzed below.

The learning of a FLNN may be considered as approximating or interpolating a continuous, multi-variate function $f(X)$ by an approximating function $f_W(X)$. Recall that in FLNN a set of basis functions Φ and a fixed number of weight parameters W are used to represent $f_W(X)$. With a specific choice of a set of basis functions, the problem is then to find the weight parameters W that provides the best possible approximation of f on the set of input–output examples. Hence, the most important point is how to choose the basis functions to obtain better approximation and is also an active research area.

Let us consider a set of basis functions $\Upsilon = \{\phi_i \in L(A)\}_{i\in I}$ with the following properties: (1) $\phi_1 = 1$, (2) the subset $\Upsilon_j = \{\phi_i \in \Upsilon\}_{i=1}^{j}$ is linearly independent set, i.e., if $\sum_{i=1}^{N}(w_i\phi_i) = 0$, then $w_i = 0$ for all $i = 1,2, \ldots, j$, and (3) $sup_j[\Sigma_{i=1}^{j}\|\phi_i\|_A^2]^{1/2} < \infty$. Let $\Upsilon_N = \{\phi\}_{i=1}^{N}$ be a set of basis functions to be considered for the FLNN. Thus, the FLNN consists of N basis functions $\{\phi_1, \phi_2, \ldots, \phi_N\} \in \Upsilon_N$ with the following input–output relationship for the jth output:

$$\hat{y}_j = \rho(s_j); \quad s_j = \sum_{i=1}^{N}(w_{ji}\phi_i(X)), \tag{6}$$

where $X \in A \subset R^n$, i.e., $X = [x_1, x_2, \ldots, x_n]^T$ is the input pattern vector, $\hat{y} \in R^m$, i.e., $\hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]^T$ is the output vector and $w_j = [w_{j1}, w_{j2}, \ldots, w_{jN}]$ is the weight vector associated with the jth output of the FLNN. The non-linear function $\rho(.) = tanh(.)$.

Considering the m-dimensional output vector, (6) can be written as

$$S = W\Phi, \tag{7}$$

where W is a $(m \times N)$ weight matrix of the FLNN given by $W = [w_1, w_2, \ldots, w_m]^T$, $\Phi = [\phi_1(X), \phi_2(X), \ldots, \phi_N(X)]^T$ is the basis function vector, and $S = [S_1, S_2, \ldots, S_N]^T$ is a matrix of linear outputs of the FLNN. the m-dimensional output vector $\widehat{y}$ may be given by

$$\widehat{y} = \rho(S) = f_W(X). \tag{8}$$

The network is trained like previous models (as discussed above) but in the present context is summarized below:

Let K patterns be applied to the network in a sequence repeatedly. Let the training sequence be denoted by (X_k, y_k) and the weight of the network be $W(k)$, where k is the iteration. Referring to (6) the jth output of the FLNN at iteration k is given by

$$\widehat{y}_j(k) = \rho\left(\sum_{i=1}^{N}(w_{ji}(k)\phi_i(X_k))\right) = \rho(w_j(k)\phi^T(X_k)), \tag{9}$$

for all $X \in A$ and $j = 1, 2, \ldots, m$, where $\phi(X_k) = [\phi_1(X_k), \phi_2(X_k), \ldots, \phi_N(X_k)]$. Let the corresponding error be denoted by $e_j(k) = y_j(k) - \widehat{y}_j(k)$.

Using the BP algorithm for a single layer, the update rule for all the weights of the FLNN is given by

$$W(k+1) = W(k) + \mu\delta(k)\phi(X_k), \tag{10}$$

where $(W(k))_m \times N$ weight matrix of the FLNN, δ and μ are the error and learning rate, respectively.

Let us discuss the motivations behind trigonometric polynomial as the basis functions of the FLNN. Without loss of generality, for all the polynomials of Nth order with respect to an orthonormal system $\{\phi_i(u)\}_{i=1}^{N}$ the best approximation in the metric space L^2 is given by the Nth partial sum of its fourier series with respect to the system. Thus, the trigonometric polynomial basis functions provide a compact representation of the function in the mean square sense. However, when the outer product terms were used along with the trigonometric polynomials for functional expansion, better results were obtained in the case of learning the function [120].

2.4 Functional link neural networks with Chebyshev polynomial

It is well known that non-linear approximation capacity of the Chebyshev orthogonal polynomial is very powerful by the best approximation theory [123]. Combining the characteristics of the FLNN and Chebyshev orthogonal polynomial resulted in a new FLNN named Chebyshev FLNN (CFLNN) [105, 106, 124]. The basic principles of this method is same as previously discussed model, but the basis function considered here are Chebyshev polynomials. The details of this model is discussed in Sect. 4, as the proposed learning scheme is used in this model for classifying the unknown pattern.

3 Functional link neural networks: a road map

FLNNs are higher-order neural networks without hidden units introduced by Klassen and Pao [125] in 1988. Despite their linear nature, FLNNs can capture non-linear input–output relationships, provided that they are fed with an adequate set of polynomial inputs, or the functions might be a subset of a complete set of orthonormal basis functions spanning an n-dimensional representation space, which are constructed out of the original input attributes [126].

In contrast to 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 the argument. Thus class separability is possible in the enhanced feature space. For example, in an exclusive-OR problem which is a non-linearly separable problem consisting of four patterns, {(0,0), (0,1), (1,0), (1,1)}, the corresponding enhanced patterns with the tensor representation of these patterns are {(0,0,0), (0,1,0), (1,0,0), (1,1,1)}. It can be observed that an exclusive-OR problem is thus transformed as a linearly separable problem.

Let us consider a two-dimensional input sample $x = [x_1, x_2]$. This sample has been mapped to a higher dimensional space by functional expansion using polynomials with certain degrees. For example, two attributes yield six polynomials up to degree 2, (i.e., $(1, x_1, x_2, x_1^2, x_2^2, x_1.x_2)$). In general, for a D-dimensional classification problem there are $\frac{(D+r)!}{D!.r!}$ possible polynomials up to degree r. For most of the real life problems, this is too big number, even for degree 2, which obviously discourages us to achieving our goal. However, we can still resort to constructive and pruning algorithms in order to address this problem. In fact Sierra et al. [84] has proposed a new algorithm for the evolution of functional link networks (EFLN) which makes use of a standard GAs [127] to evolve near minimal linear architectures. Moreover, the complexity of the algorithm still needs to be investigated.

However, the 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. FLNN with a trigonometric basis functions for classification (FLANN), as proposed in [51] is obviously an example. Note that Chebyshev FLNN is also another improvement in this direction, the detailed is discussed in Sect. 4.

Let us survey some of the potential contributions towards FLNNs and their successful applications in variety of problems.

Pao et al. [54] has presented a functional link neural network (CoFLNN) to learn the control systems. The reported results have several beneficial properties than generalized delta rule net with hidden layer and BP learning.

Haring et al. [81] has proposed an algorithm (ClFLNN) that uses evolutionary computation (specifically genetic algorithm and genetic programming) for the determination of functional links (one based on polynomials and another based on expression tree) in neural network. Patra et al. [86] has proposed a CeFLNN and applied to the problem of channel equalization in a digital communication channel. It relies on BP-learning algorithm.

Haring et al. [88] has proposed a ClaFLNN for different ways to select and transform features using evolutionary computation and shows that this kind of selection of features is a special case of so-called functional links.

Hussain et al. [102] has described a new approach for the decision feedback equalizer (DFE) based on the functional-link neural network (DfFLNN). The structure is applied to the problem of adaptive equalization in the presence of intersymbol interference (ISI), additive white Gaussian noise, and co-channel interference (CCI). The experimental results provide significantly superior bit-error rate (BER) performance characteristics compared to the conventional methods.

Chen et al. [100] has presented an adaptive implementation of the functional-link neural network (AFLNN) architecture together with a supervised learning algorithm named Rank-Expansion with Instant Learning (REIL) that rapidly determines the weights of the network. The beauty of their proposed algorithm is one-shot training as opposed to iterative training algorithms in the literature.

Dash et al. [93] has proposed a ElfFLNN with trigonometric basis functions to forecast the short-term electric load. Panagiotopoulos et al. [82] has reported better results by applying FLNN for planning in an interactive environment between two systems: the challenger and the responder. Patra et al. [83] has proposed a FLNN with BP learning (SiFLNN) for identification of non-linear dynamic systems. Moreover, Patra et al. [103] has used FLNN to adaptive channel equalization in a digital communication system with 4-QAM signal constellation named as QsFLNN. They compared the performance of the FLNN with a multilayer perceptron (MLP) and a polynomial perceptron network (PPN) along with a conventional linear LMS-based equalizer for different linear and nonlinear channel models. Out of the three ANN equalizer structures, the performance of the FLANN is found to be the best in terms of MSE level, convergence rate, BER and computational complexity for linear as well as nonlinear channel models over a wide range of SNR and EVR variations.

With the encouraging performance of FLNN [83, 86, 103], Patra et al. [89] further motivated and came up with another FLNN known as IpFLNN with three sets of basis functions such as Chebyshev, Legendre and power series to develop an intelligent model of the CPS involving less computational complexity. In the sequel, its implementation can be economical and robust.

Park and Pao [98] has reported the performance of a holistic-styled word-based approach to off-line recognition of English language script. Authors combined the practices of radial basis function neural net (RBNN) and the random-vector functional-link net approach (RVFLN) and obtained a method called the density-based random-vector functional-link net (DBRVFLN) and it is helpful in improving the performance of the word recognition.

In [84] a genetic algorithm for selecting an appropriate number of polynomials as a functional input to the network has been proposed by Sierra et al. and applied to the classification problem. However their main concern was selection of optimal set of functional links to construct the classifier. In contrast, the proposed method gives much emphasis on how to develop the learning skill of the classifier.

A Chebyshev functional link artificial neural networks (CFLNN) has proposed by Patra et al. [106] for non-linear 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 obviously an advancement in this direction. Sing et al. [95] has estimated the degree of insecurity in a power system by the proposed IeFLNN with a set of orthonormal trigonometric basis functions.

In [85], an evolutionary search of genetic type and multi-objective optimization such as accuracy and complexity of the FLNN in the Pareto sense is used to design a generalized FLNN (SyFLNN) with internal dynamics and applied to system identification.

A reduced-decision feedback functional link artificial neural network (RDF-FLNN) structure for the design of a nonlinear channel equaliser in digital communication systems is proposed by Weng et al. [101]. Authors reported that the use of direct decision feedback can greatly improve the performance of FLNN structures.

Majhi et al. [91] has applied FLNN for digital watermarking (DwFLNN), their results shows that FLNN has better performance than other algorithms in this line. In

[96], a comparative performance of three artificial neural networks has given for the detection and classification of gear faults. Authors reported that ClasFLNN is comparatively better than others.

Misra and Dehuri [51] has used a FLANN for classification problem in data mining with a hope to get a compact classifier with less computational complexity and faster learning. Purwar et al. [104] has proposed a Chebyshev functional link neural network (SyiFLNN) for system identification of unknown dynamic non-linear discrete time systems. Weng et al. [105] has proposed a reduced decision feed-back Chebyshev functional link artificial neural networks (RDF-CFLNN) for channel equalization.

Hu and Tseng [97] in 2007 has used the functional link net known as BpFLNN for classification of bankruptcy prediction.

Two simple modified CcFLNNs are proposed by Krishnaiah et al. [94] 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 [92], a FLNN with trigonometric polynomial functions (IsFLNN) are used in intelligent sensors for harsh environment that effectively linearizes the response characteristics, compensates for nonidealities and calibrates automatically. Dehuri et al. [90] has proposed a novel strategy for feature selection using genetic algorithm and then used as the input in FLNN for classification (GFLNN).

Interval regression analysis has been a useful tool for dealing with uncertain and imprecise data. Since the available data often contain outliers, robust methods for interval regression analysis are necessary. Hu [99] has proposed a genetic-algorithm-based method (IraFLNN) for determining two functional-link nets for the robust nonlinear interval regression model: one for identifying the upper bound of data interval, and the other for identifying the lower bound of data interval. Hu demonstrated that that IraFLNN performs well for contaminated data sets by resisting outliers and including all regular data in the data intervals.

With this comprehensive survey on functional link artificial neural networks, we can conclude that a very few applications has so far been made in classification task of data mining. Although theoretically this area is rich, application in classification is poor. Therefore, the proposed work can be another improvement in this direction.

Pitfalls and abuses in the functional link neural network research are harmful to the field. Some of the common misuses are: (1) mistakes in estimation of misclassification probabilities; (2) fitting of implausible functions; (3) incorrectly describing and comparing the complexity of a network; (4) no information on complexity of the network; (5) use of inadequate statistical competitors; (6) insufficient comparison with statistical method; (7) incorrect choice of basis functions; and (8) naive application to survival data. Therefore, in order for the field to grow in a healthy direction and achieve significant advances in the future, it is important for researchers to be aware of potential pitfalls and misuses as well as ways to avoid them.

Another reason that many inappropriate uses of FLANNs are published is the lack of details on several important aspects of the model-building process. Researchers often do not give sufficient detail, essential characteristics, or adequate description of their methodology, which hinders the comprehensibility or replications for others.

Tables 1 and 2 presents a summary of the development of functional link neural networks thus far and their applications in various fields.

From Tables 1 and 2, we can observed that BP-learning scheme is the mostly dominated one. However, it is associated with lots of problems, already discussed in Sect. 1. In order to alleviate some of the problems, this contribution can act as a suitable guide.

4 Proposed learning scheme for CFLNN

This section is divided into four subsections. In Sect. 4.1, the orthonormal basis of CFLNN is illustrated. In Sect. 4.2, we give a in-depth treatment of PSO with adaptive inertia named as aPSO. In Sect. 4.3, the combined effort of aPSO and BP-learning for CFLNN is discussed. Finally, we summarize CFLNN algorithm with a few computational steps added with a new learning scheme in Sect. 4.4.

4.1 Orthonormal basis of CFLNN

It is well known that the non-linear approximation capacity of the Chebyshev orthogonal polynomial is very powerful by the approximation theory [123]. Combining the characteristics of the FLNN [126] and Chebyshev orthogonal polynomial resulted in the Chebyshev functional link neural network which we named as CFLNN . The proposed method utilizes the FLNN input–output pattern, the nonlinear approximation capabilities of Chebyshev orthogonal polynomial and the adaptive particle swarm optimization (aPSO)-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

Table 3 Recursive formulae of Chebyshev polynomials

Chebyshev recurrence relation
$\varphi_1(x) = Ch_0(x) = 1$
$\varphi_2(x) = Ch_1(x) = x$
$\varphi_n(x) = Ch_{n+1}(x) = 2xCh_n(x) - Ch_{n-1}(x)$

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 aPSO–BP learning.

Recall that, 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 employe 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 convergence rapidly than expansion in other set of polynomials [123]. Moreover, Chebyshev polynomials are easier to compute than trigonometric polynomials. With these interesting properties of Chebyshev polynomial, we motivated to use CFLNN for approximation of decision boundaries in the feature space.

The first few Chebyshev polynomials are given by $Ch_0(x) = 1$, $Ch_1(x) = x$, and $Ch_2(x) = 2x^2 - 1$. The higher order Chebyshev polynomials can be generated with the recursive formulae given in Table 3.

For example, consider a two-dimensional input pattern $X = [x_1, x_2]^T$. An expanded pattern obtained by using Chebyshev functions is given by: $\varphi = [1, Ch_1(x_1), Ch_2(x_1), \ldots; 1, Ch_1(x_2), Ch_2(x_2), \ldots]^T$, where $Ch_i(x_j)$ is a Chebyshev polynomial, i the order of the polynomials choosen and $j = 1,2$.

The following theorem can guide us the cohesiveness property of CFLNN with feed forward multi-layer perceptron (MLP).

Theorem Assume a feed-forward MLP neural network with only one hidden layer and activation function of the output layer are all linear. If all the activation functions of the hidden layer satisfy the Riemann integrable condition, then the feed-forward neural network can always be represented as a Chebyshev neural network. The detailed proof of the theorem can be obtained in [123].

4.2 Adaptive particle swarm optimization (aPSO)

Adaptive particle swarm optimization (ePSO) is an improvement over the PSO [110]. PSO [128] 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:

$$S = \{x_1^k, x_2^k, \ldots, x_N^k\}$$

and a set of corresponding velocities

$$V = \{v_1^k, v_2^k, \ldots, v_N^k\}.$$

Initially, the iteration counter $k = 0$, and the positions x_i^0 and their corresponding velocities $v_i^0 (i = 1, 2, \ldots, N)$, 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 ($i = 1, 2, \ldots, N$) is biased towards its best position p_i^k with best function value 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

$$P = \{p_1^k, p_2^k, \ldots, p_N^k\},$$

where $p_i^0 = x_i^0, \forall i$.

We can say a particle in S 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 S the worst (the best) if $p_i^k (p_j^k)$ is the least (best) fitted, with respect to function value in P. We denote the *pbest* of the worst particle and the best particle in S as p_h^k and p_g^k, respectively. Hence $p_g^k = \text{argmin}_{i \in 1,2,\ldots,N} f(p_i^k)$ and $p_h^k = \text{argmax}_{i \in 1,2,\ldots,N} f(p_i^k)$.

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 S is updated for the next iteration using

$$x_i^{k+1} = x_i^k + v_i^{k+1}, \tag{11}$$

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

The parameters r_1 and r_2 are uniformly distributed random numbers in [0, 1] and c_1 and c_2, known as the cognitive and social parameters, respectively, are popularly chosen to be $c_1 = c_2 = 2.0$ [128]. 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} are created using the following rules with a minimization of the cost function: $p_i^{k+1} = x_i^{k+1}$ if $f(x_i^{k+1}) < f(p_i^k)$, otherwise $p_i^{k+1} = p_i^k$.

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 [129] has done the first modification by introducing a constant inertia w, 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

$$v_i^{k+1} = w \cdot v_i^k + r_1 \cdot c_1 \cdot (p_i^k - x_i^k) + r_2 \cdot c_2 \cdot (p_g^k - x_i^k), \quad (12)$$

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

Again Shi and Eberhart [130] 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 τ ($0 < \tau \leq 1$) of the distance between the bounds of the search space, i.e., $v^{\max} = \tau(x^u - x^l)$.

Forie and Groenwold [131] 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 pre-specified number of iterations h, i.e.,

if $f(p_g^k) = f(p_g^{k-1})$ then $w_{k+1} = \alpha w_k$ and $v_k^{\max} = \beta v_k^{\max}$, where α and β are such that $0 < \alpha, \beta < 1$.

Clerc and Kennedy [132] 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 progresses. In this modification, the velocity update is given by $v_i^{k+1} = \chi\ (v_i^k + r_1 c_1 (p_i^k) + r_2 c_2 (p_g^k - x_i^k))$, where $\chi = \frac{2}{|2-\phi-\sqrt{\phi^2-4\phi}|}$, $\phi = c_1 + c_2 > 4$.

Da and Ge [118], also modified PSO by introducing a temperature-like control parameter as in the simulated annealing algorithm. Zhang et al. [133] has modified the PSO by introducing a new inertia weight during the velocity update. Generally in the beginning stages of their algorithm, the inertial weight w, should be reduce rapidly, when around optimum the inertial weight w should be reduced slowly. They adopted the following rule:

$$w = w_0 - \left(\frac{w_1}{MAXITER1}\right) \times k, \quad \text{if } 1 \leq k \leq MAXITER1,$$
$$\text{and } w = (w_0 - w_1) \times \exp((MAXITER1 - k)/v),$$
$$\text{if } MAXITER1 < k \leq MAXITER,$$

where w_0 is the initial inertia weight, w_1 is the inertial weight of linear section ending, *MAXITER* are the total searching generations; *MAXITER*1 are the used generations that inertia weight is reduced linearly, k is is a variable whose range is [1, *MAXITER*]. By adjusting k, different ending values of inertial weight are obtained.

In this learning, the inertial weight is adapted as a part of searching the optimal sets of weights. Compared to [130, 131], in our adaptive PSO, the particle position is adjusted such that the highly fit particle (best particle) moves slowly when compared to the less fit particle. This can be achieved by selecting different w values for each particle according to their rank, between w^u and w^l as in the following form:

$$w_i = w^l + \left(\frac{\text{Rank}_i}{\text{SWARM\_SIZE}(N)}\right) \times (w^u - w^l). \quad (13)$$

Therefore, from (13), it can be seen that the best particle takes the first rank, and the inertia weight for that particle is set to the minimum value while that for the lowest fitted particle takes the maximum inertia weight, which makes that particle move with a high velocity.

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 [134].

$$c_1^k = (c_{1f} - c_{1i}) \frac{k}{MAXITER} + c_{1i}, \quad (14)$$

and

$$c_2^k = (c_{2f} - c_{2i}) \frac{k}{MAXITER} + c_{2i}, \quad (15)$$

4.3 aPSO–BP learning algorithm

The aPSO -BP is a learning algorithm which combines the best attributes of aPSO with the best attribute of BP algorithm. The aPSO algorithm is a global algorithm, which has a strong ability to find global optimistic result. This aPSO 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 aPSO with the BP, a new algorithm referred to as aPSO–BP hybrid learning algorithm is formulated as a part of 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 change is smaller than a predefined number, the searching process is switched to gradient descending searching according to this heuristic knowledge. Similar to the aPSO algorithm, the aPSO–BP

algorithm's searching process is also started from initializing a group of random particles. First, all the particles are updated according to the (5), 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 aPSO–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].
2. Get the input parameters: *MAXITER*, w^l, w^u, c_{1i}, c_{2i}, c_{1f}, and c_{2f}.
3. Evaluate each initialized particle's fitness value, and p_b is set as the positions of the current particles, while p_g is set as the best position of the initialized particles.
4. If the maximal iterative generations are arrived, go to step 11, else, go to step 5.
5. Evaluate the inertia factor according to (13), so that each particles movement is directly controlled by its fitness value.
6. Adjust the value of c_1 and c_2 by using (14) and (15).
7. The positions and velocities of all the particles are updated according to (11), then a group of new particles are generated.
8. Evaluate each new particle's fitness value, and the worst particle is replaced by the stored best particle. If the ith particle's new position is better than p_b, p_b is set as the new position of the ith particle. If the best position of all new particles are better than p_g, then p_g is updated.
9. If the current p_g is unchanged for 15 consecutive generations, then go to step 9; else, go to step 4.
10. 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.
11. Output 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:

$$\mu = k\exp(-v \times epoch), \tag{16}$$

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

4.4 aPSO–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 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 [51].

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

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

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 $\widehat{y}_i(t) = \rho(\Sigma_{j=1}^{D'}\psi_j(x_{ij}) \cdot w_{ij})\forall i$. The error associated with the ith output is given by $e_i(t) = y_i(t) - \widehat{y}_i(t)$. 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 aPSO–BP learning rules and go to step 3.

5 Experimental details

This is divided into five subsections. Sect. 5.1 describes the datasets taken from UCI [119] repository of machine learning databases. The parameters required for the proposed method are given in Sect. 5.2. The performance of the CFLNN with aPSO–BP learning using some of the datasets especially considered by Sierra et al. [84] compared with the model proposed by Sierra et al. in Sect. 5.3. In Sect. 5.4, the classification accuracy of CFLNN with aPSO–BP learning is compared with FLNN [51]. In Sect. 5.5, we compared the performance of CFLNN with aPSO–BP learning with FLNN proposed in [51] using the cost matrix analysis and then compared with the results obtained by StatLog project [135]. Hereafter, we refer CFLNN with aPSO–BP learning algorithm as HCFLNN.

5.1 Description of the datasets

The availability of results, with previous evolutionary and constructive algorithms (e.g., Sierra et al. [84], Preshelt [136]) has guided us in the selection of the following varied datasets taken from the UCI repository of machine learning databases for the addressed neural network learning. Let us briefly discuss the datasets, used for our experimental setup.

IRIS dataset The dataset consists of $d = 4$ features made on each of the 150 iris plants of class $c = 3$ species. The three distinct species corresponds to three different classes such as Iris Setosa, Iris Versicolor and Iris Virginica. The problem is to classify each test point to its correct species based on the four measurements.

WINE dataset These data are the results of a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The 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 13; all are continuous in nature. There are no missing attributes.

PIMA Indians diabetes dataset This dataset consists of $d = 8$ numerical medical attributes and $c = 2$ classes (tested positive or negative for diabetes). There are $n = 768$ instances. Further, it has dataset related to the diagnosis of diabetes in an Indian population that lives near the city of Phoenix, Arizona. All inputs are continuous; 65.1% samples are diabetes negatives.

BUPA liver disorders dataset This is related to the diagnosis of liver disorders and created by BUPA Medical Research, Ltd. It consists of 6 numerical attributes, 345 patterns and 2 classes.

Heart disease dataset This is related to diagnoses of people with heart problems. It has 270 patterns, 6 attributes and 2 classes.

Cancer dataset In this dataset, the task of classifier is to classify a tumor as either benign or malignant based on cell descriptions gathered by microscopic examinations. Input attributes are for instance the clump thickness, the uniformity of cell size and cell shape, the amount of marginal adhesion, and the frequency of bare nuclei. It is described by 9 inputs, 2 classes, and 699 samples. All inputs are continuous, 34.5% belong to the class of malignant.

5.2 Parameters

All the algorithms have some parameters that have to be provided by the user. The parameters for the proposed HCFLNN is listed in Table 4 based on the experimental results obtained under several independent runs. However, the parameters for other algorithms are set based on suggestion. Even though no systematic parameter optimization process has so far been attempted, the suggested one (producing significantly better than our own setting specifically in the case of EFLN). The parameters for EFLN were adopted as suggested in [84]. Similarly, the parameters for FLANN was set as suggested in [51].

The values of the parameters used in this paper are as follows: we set $N = 20d$, where d is the dimension of the problem under consideration. The upper limit (w^u) and lower limit (w^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 v in HCFLNN was chosen after

Table 4 Description of the parameters

Symbol	Purpose of the symbol
N	Size of the swarm
w	Inertia weight
w^u	Upper limit of the inertia
w^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

Table 5 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

several runs to obtain the best results. In a similar manner, the functional expansion of the HCFLNN was carried out.

5.3 HCFLNN versus EFLN

In this subsection we will compare the results of HCFLNN with the results of EFLN with polynomial basis functions of degrees 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. [84] has given some guidance to optimize the polynomial degree that can best suit the architecture. Considering degrees of the polynomials 1, 2 and 3, the possible number of expanded inputs of the above datasets are given in Table 5.

For the sake of convenience, we report the results of the experiments conducted on CANCER and BUPA and then compared with the methods EFLN [84]. We partitioned both the datasets into three sets: training, validation and test sets. Both the networks are trained for 1,500 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 1,500 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 6.

Table 6 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

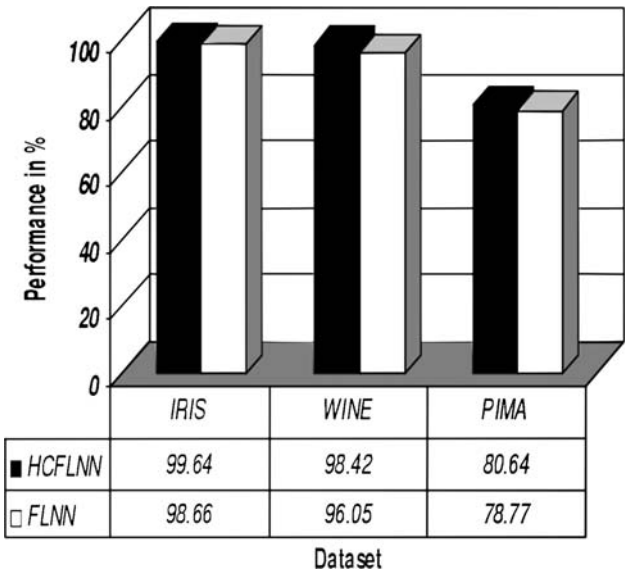

Fig. 2 Classification accuracy in training set

5.4 HCFLNN versus FLANN

Here, we will discuss the comparative performance of HCFLNN with FLANN using three datasets IRIS, WINE, and PIMA. In this case, the total set of samples is randomly divided into two equal folds. Each of these two folds are

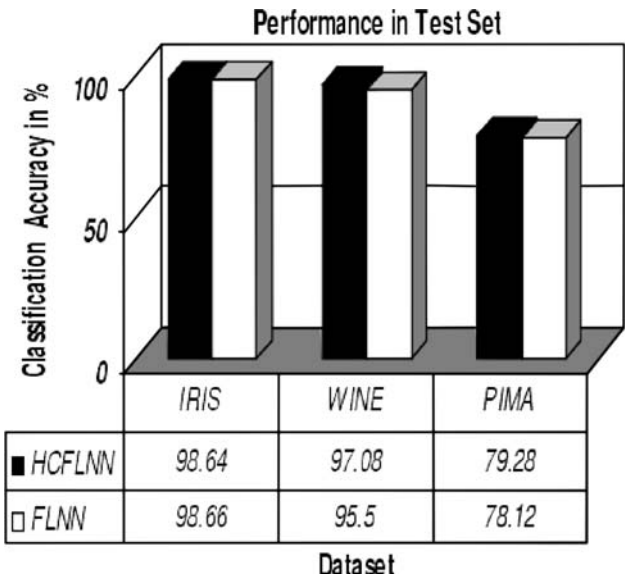

Fig. 3 Classification accuracy in test set

Table 7 Comparative average performance of HCFLNN and FLNN [51] based on the confidence level ($\alpha = 95\%$)

Dataset	HCFLNN	FLANN
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 8 Comparative average performance of HCFLNN and FLANN [51] based on the confidence level ($\alpha = 98\%$)

Dataset	HCFLNN	FLANN
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

alternatively used either as a training set or as a test set. As the proposed learning method aPSO–BP learning is a stochastic algorithm, hence, 10 independent runs were performed for every single fold. The training results obtained in the case of HCFLNN, averaged over 10 runs are illustrated in Fig. 2 and compared with the single run of FLANN. Similarly, Fig. 3 illustrates the performance of both classifiers in test set.

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

The comparative performance of HCFLNN with FLANN [51] is given in the Tables 7, and 8 w.r.t to the different confidence level (α) of 95, and 98%, respectively. In the case of PIMA, the performance of HCFLNN and FLANN in test data is competitive with each other, whereas a clear edge is obtained in training set. The classification accuracy of HCFLNN in the training cases of IRIS data is better than FLANN, however, in the test cases the classification accuracy is competitive.

In the case of WINE data the training and testing performance of HCFLNN is superior compared to FLANN.

5.5 Performance of HCFLNN versus FLANN 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 ninefold cross-validation methodology. The reason for using ninefold cross validation is that to compare the performance with the performance of few of the representative algorithms considered in StatLog Project [135]. In ninefold 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).

Table 9 Weight matrix of classes to penalize

Real classification	Model classification	
	Class 1	Class 2
Class 1	0	w_2
Class 2	w_1	0

Table 10 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

The procedure makes use of a weight matrix, which is described in Table 9.

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 w_1, while the weight of the penalty for class 1 records that are classified as class 2 samples is w_2. Therefore, the metric used for measuring the cost of the wrongly classifying patterns in the training and test dataset is given by (17) and (18).

$$C_{\text{train}} = (S_1 \times w_1 + S_2 \times w_2)/S_{\text{train}}, \tag{17}$$

$$C_{\text{test}} = (S_1 \times w_1 + S_2 \times w_2)/S_{\text{test}}, \tag{18}$$

Table 11 Heart disease classification performance of HCFLNN 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

Table 12 Comparative classification performance of HCFLNN, FLANN with the algorithms considered in [135] using the heart disease bench mark datset

Methods	C_{test}	C_{train}
HCFLNN	**0.31**	**0.32**
FLANN	0.33	0.34
$HNFB^{-1}$	0.37	0.59
Bayes	0.37	0.35
Dicrim	0.39	0.31
LogDisc	0.39	0.27
Alloc80	0.41	0.39
QuaDisc	0.42	0.27
Castle	0.44	0.37
Cal5	0.44	0.33
CART	0.45	0.44
CASCADE	0.47	0.21
Knn	0.478	0
Smart	0.48	0.26
Dipole92	0.51	0.43
Itrule	0.51	–
BayTree	0.53	0.11
LVQ	0.60	0.14
IndCart	0.63	0.26
Kohonen	0.69	0.43
Ac2	0.74	0
Cn2	0.77	0.21
C4.5	0.78	0.44

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 belongs to class 1 and 2, respectively; S_{train} and S_{test} are the total number of training and test patterns, respectively.

Table 10 presents the errors and costs of the training and test sets for the FLANN model with a weight value of $w_1 = 5$ and $w_2 = 1$.

Table 11 illustrates the performance of HCFLNN 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 [135]. According to [135], comparison consists of calculating the average cost produced by the nine data subsets used for validation. Table 12 presents the average cost for the nine training and test subsets. The result of the HCFLNN is highlighted in bold.

6 Conclusions and research directions

Functional link neural networks is very young, but active and rapidly expanding field of research. In other words, it is an active branch of higher order neural networks (HONs) and is widely used in many applications such as control, channel equalization, bankruptcy prediction, electric load forecasting, data mining, and so on. The promise of FLNN is to provide a very simple and compact architecture (possible no hidden layers) for approximation of highly non-linear boundary.

In this survey, we have discussed the prospective models of FLNN with their associated basis functions and learning scheme followed by a clear road map of the development of FLNNs over the years by several researchers. In addition, we developed a new learning scheme for Chebyshev functional link neural network (CFLNN). The model is constructed using the newly proposed aPSO-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 the case of IRIS data. Further, we compared this model with EFLN and FLNN, respectively. The comparative results of the developed model is shows 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.

With this encouraging results of HCFLNN our future research includes: (1) testing the proposed method on more number of real life bench mark classification problems with highly non-linear boundaries; (2) mapping the input

features with other polynomials such as Legendre, Gaussian, Sigmoid, power series, etc. for better approximation of the decision boundaries; (3) the stability and convergence analysis of the proposed method; and (4) the evolution of optimal FLNN using particle swarm optimization. Of course simple methods perform well and often better than more complex approaches but innovative combinations of FLNN with PSO are still promising.

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 [137, 138] such as clustering, feature selection, feature extraction, association rule mining, regression, and so on. However, a necessary property of algorithms which are capable of handling large and growing datasets is their scalability or linear complexity with respect to the data size. Scalability in the data mining literature means a time (and space) complexity which is proportional to the size of the data set, i.e., $O(n)$ if n is the number of records of a data set. The proportionality *constant* may actually grow slightly as well and complexities like $O(n \log(n))$ are usually also acceptable. Moreover, the variables, or attributes, are mainly assumed to be either continuous or categorical but more general data types are frequently analysed in data mining.

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.

Acknowledgments Authors would like to thank BK21 research program on Next Generation Mobile Software at Yonsei University, South Korea for their financial support. The authors greatly appreciate all the reviewers' constructive comments that motivated them to think more and improve the presentation of this paper.

References

1. Haykin S (1999) Neural networks—a comprehensive foundation. Prentice Hall, Englewood Cliffs
2. Zhang GP (207) Avoiding pitfalls in neural network research. IEEE Trans Syst Man Cybern Part C Appl Rev 37(1):3–16
3. Anders U, Korn O (1999) Model selection in neural networks. Neural Netw 12:309–323
4. Benitez JM, Castro JL, Requena I (1997) Are artificial neural networks black boxes? IEEE Trans Neural Netw 8(5):1156–1164
5. Cheng B, Titterington D (1994) Neural networks: a review from a statistical perspective. Stat Sci 9(1):2–54
6. McCulloch W, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 7:115–133
7. Giles CL, Maxwell T (1987) Learning invariance, and generalization in a higher order neural networks. Appl Opt 26(23):4972–4978
8. Belli MR, Conti M, Crippa P, Turchetti C (1999) Artificial neural networks as approximators of stochastic processes. Neural Netw 12(4–5):647–658
9. Castro JL, Mantas CJ, Benitez JM (2000) Neural networks with a continuous squashing function in the output are universal approximators. Neural Netw 13(6):561–563
10. Funahashi K (1989) On the approximate realization of continuous mappings by neural networks. Neural Netw 2:183–192
11. Andrews R, Diederich J, Tickle AB (1995) Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl Based Syst 8(6):373–389
12. Castro JL, Requena I, Benitez JM (2002) Interpretation of artificial neural networks by means of fuzzy rules. IEEE Trans Neural Netw 13(1):101–116
13. Setiono R, Leow WK, Zurada J (2002) Extraction of rules from artificial neural networks for nonlinear regression. IEEE Trans Neural Network 13(3):564–577
14. Setiono R, Thong JYL (2004) An approach to generate rules from neural networks for regression problems. Eur J Oper Res 155:239–250
15. Gish H (1990) A probabilistic approach to the understanding and training of neural network classifiers. In: Proc IEEE international conference acoustic, speech signal process 3:1361–1364
16. Zhang GP (2000) Neural networks for classification: a survey. IEEE Trans Syst Man Cybern C 30(4):451–462
17. Michie D, Spiegelhalter DJ, Taylor CC (1994) Machine learning, neural and statistical classification. Ellis Horwood, New York
18. Adya M, Collopy F (1998) How effective are neural networks at forecasting and prediction? A review and evaluation. J Forecast 17:481–495
19. Callen JL, Kwan CCY, Yip PCY, Yuan Y (1996) Neural network forecasting of quarterly accounting earnings. Int J Forecast 12:475–482
20. Church KB, Curram SP (1996) Forecasting comsumers' expenditure: a comparison between econometric and neural network models. Int J Forecast 12:255–267
21. Connor JT, Martin RD, Atlas LE (1994) Recurrent neural networks and robust time series prediction. IEEE Trans Neural Netw 51(2):240–254
22. Cottrell M, Girard B, Girard Y, Mangeas M, Muller C (1995) Neural modeling for time series: a statistical stepwise method for weight elimination. IEEE Trans Neural Netw 6(6):1355–1364
23. Faraway JJ, Chatfield C (1998) Time series forecasting with neural networks: a comparative study using the airline data. Appl Stat 47:231–250
24. Fletcher D, Goss E (1993) Forecasting with neural networks—an application using bankruptcy data. Inf Manag 24:159–167
25. Gorr WL (1994) Research prospective on neural network forecasting. Int J Forecast 10:1–4
26. Hippert HS, Pedreira CE, Souza RC (2001) Neural networks for short-term forecasting: a review and evaluation. IEEE Trans Power Syst 16(1):44–55
27. Hu MY, Zhang GP, Jiang CX, Patuwo BE (1999) A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting. Decis Sci 30:197–216
28. Kaastra I, Boyd M (1996) Designing a neural network for forecasting financial and economic time series. Neurocomputing 10:215–236
29. Maier HR, Dandy GC (2000) Neural networks for the prediction and forecasting of water resources variables: a review of modeling issues and applications. Environ Model Softw 15:101–124
30. Park YR, Murray TJ, Chen C (1996) Predicting sun spots using a layered perceptron neural network. IEEE Trans Neural Netw 7(2):501–505
31. Qi M, Zhang GP (2001) An investigation of model selection criteria for neural network time series forecasting. Eur J Oper Res 132:666–680

32. Kracha KA, Wagner U (1999) Applications of artificial neural networks in management science: a survey. J Retail Consum Serv 6:185–203
33. Wong BK, Bodnovich TA, Selvi Y (1997) Neural network applications in business: a review and analysis of the literature (1988–1995). Decis Support Syst 19:301–320
34. Flood I, Kartam N (1994) Neural network in civil engineering-I: principles and understanding. J Comput Civil Eng 8(2):131–148
35. Lu CN, Wu HT, Vemuri S (1993) Neural network based short term load forecasting. IEEE Trans Power Syst 8(1):336–342
36. Lisboa PJG (2002) A review of evidence of health benefit from artificial neural networks in medical intervention. Neural Netw 15:11–39
37. Protney LG, Watkins MP (2000) Foundations of clinical research: applications to practice. Prentice-Hall, Princeton
38. Hosseini-Nezhad SM, Yamashita TS, Bielefeld RA, Krug SE, Pao YH (1995) A neural network approach for the determination of interhospital transport mode. Comput Biomed Res 28(4):319–334
39. Tawfik H, Liatsis P (1997) Prediction of non-linear time series using higher order neural networks. In: Proceeding IWSSIP1997 conference, Poznan, Poland
40. Kaita T, Tomita S, Yamanaka J (2002) On a higher order neural network for distortion invariant pattern recognition. Pattern Recognit Lett 23:977–984
41. Ghosh J, Shin Y (1992) Efficient higher-order neural networks for classification and function approximation. Int J Neural Syst 3:323–350
42. Minsky M, Papert S (1969) Perceptrons. The MIT Press
43. Widrow B, Hoff ME (1960) Adaptive switching circuits. IRE WESCON Convention Record, pp 96–104
44. Widrow B, Lehr M (1990) 30 years of adaptive neural networks: perceptron, madaline, and back-propagation. Proc IEEE 78(9):1415–1442
45. Cover TM (1965) Geometrical and statistical properites of systems of linear inequalities with applications in pattern recognition. IEEE Trans Electron Comput 14:326–334
46. Hornik K et al (1989) Multi-layer feed-forward networks are universal approximators. Neural Netw 2:359–366
47. Giles CL, Maxwell T (1987) Learning, invariance and generalization in higher-order neural networks. Appl Opt 26(23):4972-4978
48. Pao YH (1989) Adaptive pattern recognition and neural network. Addison-Wesley, Reading, MA
49. Venkatesh SS, Baldi P (1991) Programmed interactions in higher order neural networks: maximal capacity. J Complex 7:316–337
50. Antyomov E, Pecht OY (2005) Modified higher order neural network for invariant pattern recognition. Pattern Recognit Lett 26:843–851
51. Misra BB, Dehuri S (2007) Functional link neural network for classification task in data mining. J Comput Sci 3(12):948–955
52. Mirea L, Marcu T (2002) System identification using functional link neural networks with dynamic structure. 15th Triennial World Congress, Barcelona, Spain
53. Cass R, Radl B (1996) Adaptive process optimization using functional link networks and evolutionary algorithms. Control Eng Pract 4(11):1579–1584
54. Pao Y-H, Philips SM (1995) The functional link net learning optimal control. Neurocomputing 9:149–164
55. Shin Y, Ghosh J (1995) Ridge polynomial networks. IEEE Trans Neural Netw 6(2):610–622
56. Shin Y, Ghosh J (1992) Approximation of multivariate functions using ridge polynomial networks. In: Proceedings of international joint conference on neural networks II, pp 380–385
57. Voutriaridis C, Boutalis YS, Mertzios G (2003) Ridge polynomial networks in pattern recognition. 4th EURASIP conference focused on video/image processing and multimedia communications, Croatia, pp 519–524
58. Shin Y, Ghosh J (1991) The pi-sigma networks: an efficient higher order neural network for pattern classification and function approximation. In: Proceedings of international joint conference on neural networks I, pp 13–18
59. Shin Y, Ghosh J (1992) Computationally efficient invariant pattern recognition with higher order pi-sigma networks. The University of Texas at Austin, Tech. Report
60. Shin Y, Ghosh J (1991) Realization of boolean functions using binary pi-sigma networks. In: Proceedings of conference on artificial neural networks in engineering, St. Louis
61. Hussain AJ, Liatsis P (2002) Recurrent pi-sigma networks for DPCM image coding. Neurocomputing 55:363–382
62. Xiong Y et al (2007) Training pi-sigma network by on-line gradient algorithm with penalty for small weight update. Neural Comput 19:3356–3368
63. Iyoda EM et al (2007) Image compression and reconstruction using pi_t-sigma neural networks. Soft Comput 11:53–61
64. Hussain AJ et al (2008) Physical time series prediction using recurrent pi-sigma neural networks. Int J Artif Intell Soft Comput 1(1):130–145
65. Nie Y, Deng W (2008) A hybrid genetic learning algorithm for pi-sigma neural network and the analysis of its convergence. In: Proceedings of fourth international conference on natural computation, IEEE Press, pp 19–23
66. Zhu Q, Cai Y, Liu L (1999) A global learning algorithm for a RBF network. Neural Netw 12:527–540
67. Li M, Tian J, Chen F (2008) Imrpoving multiclass pattern recognition with a co-evolutionary RBFNN. Pattern Recognit Lett 29:392–406
68. Dybowski R (1998) Classification of incomplete feature vectors by radial basis function networks. Pattern Recognit Lett 19:1257–1264
69. Leonardis A, Bischof H (1998) An efficient MDL based construction of RBF networks. Neural Netw 11:963–973
70. Chen S, Wu Y, Luk BL (1999) Combined genetic algorithm optimization and regularized orthogonal least square learning for radial basis function networks. IEEE Tran Neural Netw 10(5):1239–1243
71. Lee YC, Doolen G, Chen HH, Sun GZ, Maxwell T, Lee HY, Giles CL (1986) Machine learning using a higher order correlation network. Physica 22D:276–306
72. Peretto P, Niez JJ (1986) Long-term memory storage capacity of multiconnected neural networks. Biol Cybern 54:5363
73. Psaltis D, Park CH (1986) Nonlinear discriminant functions and associative memories. In: Denker JS (ed) Neural networks for computing. Amererican Institute of Physics, New York, pp 370–375
74. Gardner E (1987) Multiconnected neural-network models. J Phys A Math Gen 20:3453–3464
75. Abbott LF, Arian Y (1987) Storage capacity of generalized networks. Phys Rev A 36:5091–5094
76. Kamp Y, Hasler M (1990) Recursive neural networks for associative memory. Wiley, New York
77. Horn D, Usher M (1988) Capacities of multiconnected memory models. J Phys France 49:389–395
78. Guillermo V (1998) A distributed approach to neural network simulation program. Master thesis, The University of Texas at E1 Paso, TX
79. Zurada JM (1992) Introduction to artificial neural system. West Publishing Company, St. Paul, MN
80. Beale R, Jackson T (1991) Neural computing: an introduction. Hilger, Philadelphia, PA
81. Haring B, Kok JN (1995) Finding functional links for neural networks by evolutionary computation. In: Van de Merckt T

et al (eds) BENELEARN1995, proceedings of the fifth Belgian–Dutch conference on machine learning, Brussels, Belgium, pp 71–78
82. Panagiotopoulos DA et al (1999) Planning with a functional neural network architecture. IEEE Trans Neural Netw 10(1):115–127
83. Patra JC et al (1999) Identification of non -linear dynamic systems using functional link artificial neural networks. IEEE IEEE Trans Syst Man Cyber Part B Cybern 29(2):254–262
84. Sierra A, Macias JA, Corbacho F (2001) Evolution of Functional Link Networks. IEEE Tranas Evol Comput 5(1):54–65
85. Marcu T, Koppen-Seliger B (2004) Dynamic functional link neural networks genetically evolved applied to system identification. In: Proceedings of ESANN'2004, Bruges (Belgium), pp 115–120
86. Patra JC, Pal NR (1995) A functional link neural network for adaptive channel equalization. Signal Process 43:181–195
87. Zhao H, Zhang J (2008) Functional link neural network cascaded with Chebyshev orthogonal polynomial for non-linear channel equalization. signal Process 88:1946–1957
88. Haring et al (1997) Feature selection for neural networks through functional links found by evolutionary computation. In: Liu X et al (eds) Adavnces in intelligent data analysis (IDA-97). LNCS 1280:199–210
89. Patra JC et al (2000) Modelling of an intelligent pressure sensor using functional link artificial neural networks. ISA Trans 39:15–27
90. Dehuri S et al (2008) Genetic feature selection for optimal functional link neural network in classification. In: Fyfe C et al (eds) IDEAL 2008, LNCS 5326:156–163
91. Majhi B et al (2005) An improved scheme for digital watermarking using functional link artificial neural network. J Comput Sci 1(2):169–174
92. Patra JC et al (2008) Functional link neural networks-based intelligent sensors for Harsh Environments. Sens Transducers J 90:209–220
93. Dash PK et al (1999) A functional link neural network for short term electric load forecasting. J Intell Fuzzy Syst 7:209–221
94. Krishnaiah D et al (2008) Application of ultrasonic waves coupled with functional link neural network for estimation of carrageenan concentration. Int J Phys Sci 3(4):90–96
95. Sing SN, Srivastava KN (2002) Degree of insecurity estimation in a power system using functional link neural network. ETEP 12(5):353–359
96. Abu-Mahfouz I-A (2005) A comparative study of three artificial neural networks for the detection and classification of gear faults. Int J Gen Syst 34(3):261–277
97. Hu Y-C, Tseng F-M (2007) Functional-link net with fuzzy integral for bankruptcy prediction. Neurocomputing 70:2959–2968
98. Park GH, Pao YH (2000) Unconstrained word-based approach for off-line script recognition using density based random vector functional link net. Neurocomputing 31:45–65
99. Hu Y-C (2008) Functional link nets with genetic algorithm based learning for robust non-linear interval regression analysis. Neurocomputing. doi:10.1016/J.neucom.2008.07.002
100. Chen CLP et al (1998) An incremental adaptive implementation of functional link processing for function approximation, time series prediction, and system identification. Neurocomputing 18:11–31
101. Weng W-D, Yen CT (2004) Reduced decision feed-back FLANN non-linear channel equaliser for digital communication systems. IEE Proc Commun 151(4):305–311
102. Hussain A et al (1997) A new adaptive functional link neural network based DFE for overcoming co-channel interference. IEEE IEEE Trans Commun 45(11):1358–1362
103. Patra JC et al (1999) Non-linear channel equalization for QAM signal constellation using artificial neural networks. IEEE Tranasactions on Systems, Man, Cybernetics-Part B: Cybernetics 29(2):262–271
104. Purwar S et al (2007) On-line system identification of complex systems using Chebyshev neural networks. Appl Soft Comput 7:364–372
105. Weng W-D et al (2007) A channel equalizer usi ng reduced decision feedback Chebyshev function link artificial neural networks. Inf Sci 177:2642–2654
106. Patra JC et al (2002) Non-linear dynamic system identification using Chebyshev functional link artificial neural networks. IEEE Trans Syst Man Cybern Part B Cybern 32(4):505–511
107. Fogel DB (2000) Evolutionary computation: towards a new philosophy of machine intelligence. IEEE Press, New York
108. Pearson DW et al (eds) (1995) Artificial neural networks and genetic algorithms. Springer Verlag
109. Suzuki J (1995) A Markov chain analysis on simple genetic algorithms. IEEE Trans Syst Man Cybern 25(4):6–659
110. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. Pisacataway, NJ, pp 1942–9148
111. Schaffer JD, Whitley D, Eshelman LJ (1992) 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
112. Davidor Y (1990) Epistasis variance: suitability of a representation to genetic algorithms. Complex Syst 4:368–383
113. Eshelman LJ, Schaffer JD (1993) Real coded genetic algorithms and interval schemata. In: Whitley LD (ed) Foundation of genetic algorithms. Morgan Kaufmann, San Mateo, pp 187–202
114. Muhlenbein H, Schlierkamp-Voosen D (1993) Predictive models for the breeder genetic algorithm I. Continuous parameters optimization. Evol Comput 1(1):24–49
115. Schutte JF, Groenwold AA (2005) A study of global optimization using particle swarms. J Glob Optim 31(1):93–108
116. Ali MM, Kaelo P (2008) Improved particle swarm algorithms for global optimization. Appl Math Comput 196:578–593
117. Yu J, Wang S, Xi L (2008) Evolving artificial neural networks using an improved PSO and DPSO. Neurocomputing 71:1054–1060
118. Da Y, Ge XR (2005) An improved PSO-based ANN with simulated annealing technique. Neurocomput Lett 63:527–533
119. Blake CL, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html
120. Pao Y-H, Phillips SM, Sobajic DJ (1992) Neural-net computing and intelligent control systems. Int J Control 56(2):263–289
121. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4:251–257
122. Smith KA, Gupta JND (2002) Neural networks in business: techniques and applications. Idea Group, Hershey, PA
123. Lee TT, Jeng JT (1998) The Chebyshev polynomial based unified model neural networks for function approximations. IEEE Trans Syst Man Cybern Part B 28:925–935
124. Namatame A, Veda N (1992) Pattern classification with Chebyshev neural network. Int J Neural Netw 3:23–31
125. Klasser MS, Pao YH (1988) Characteristics of the functional link net: a higher order delta rule net. IEEE proceedings of 2nd annual international conference on neural networks, San Diago, CA
126. Pao YH, Takefuji Y (1992) Functional link net computing: theory, system, architecture and functionalities. IEEE Comput, pp 76–79
127. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Morgan Kaufmann, San Mateo
128. Kennedy J, Eberhart RC (1999) The particle swarm: social adaptation in information processing systems. In: Corne D,

Dorigo M, Glover F (eds) New ideas in optimization. McGraw–Hill, Cambridge, UK, pp 379–387
129. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of the IEEE international conference on evolutionary computation. IEEE Press, Pisacataway, NJ, pp 69–73
130. Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. Evolutionary Programming VII, LNCS, Springer, Berlin 1447:591–600
131. Forie PC, Groenwold AA (2002) The particle swarm optimization algorithm in size and shape optimization. Struct Multidiscipl Optim 23(4):259–267
132. Clerc M, Kennedy J (2002) The particle swarm explosion, stability and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73
133. Zhang JR et al (2007) A hybrid particle swarm optimization-back-propagation algorithm for feed-forward neural network training. Appl Math Comput 185:1026–1037
134. Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255
135. Lippmann R (1987) An introduction to computing with neural networks. IEEE ASSP Mag 4:4–22
136. Preshelt L (1994) Proben1-a set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Universitat Karlsruhe, Germany
137. Ghosh A, Dehuri S, Ghosh S (2008) Multi-objective evolutionary algorithms for knowledge discovery from databases. Springer
138. Kriegel H-P et al (2007) Future trends in data mining. Data Mining Knowl Discov 15(1):87–97
139. Vellido A, Lisboa PJG, Vaughan J (1999) Neural networks in business: a survey of applications (1992–1998). Expert Syst Appl 17:51–70
140. Liatsis P, Hussain AJ (1999) Non-linear one dimensional DPCM image prediction using polynomial neural network. In: Proceedings of SPIE applications of artificial neural networks in image processing IV, San Jose, CA 3647:58–68