Engineering Applications of Artificial Intelligence 60 (2017) 97–116

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Metaheuristic design of feedforward neural networks: A review of two decades of research

Varun Kumar Ojha[a,*], Ajith Abraham[b], Václav Snášel[a]

[a] Department of Computer Science, VŠB-Technical University of Ostrava, Ostrava, Czech Republic
[b] Machine Intelligence Research Labs (MIR Labs), Auburn, WA, USA

ARTICLE INFO

Keywords:
Feedforward neural network
Metaheuristics
Nature-inspired algorithms
Multiobjective
Ensemble

ABSTRACT

Over the past two decades, the feedforward neural network (FNN) optimization has been a key interest among the researchers and practitioners of multiple disciplines. The FNN optimization is often viewed from the various perspectives: the optimization of weights, network architecture, activation nodes, learning parameters, learning environment, etc. Researchers adopted such different viewpoints mainly to improve the FNN's generalization ability. The gradient-descent algorithm such as backpropagation has been widely applied to optimize the FNNs. Its success is evident from the FNN's application to numerous real-world problems. However, due to the limitations of the gradient-based optimization methods, the metaheuristic algorithms including the evolutionary algorithms, swarm intelligence, etc., are still being widely explored by the researchers aiming to obtain generalized FNN for a given problem. This article attempts to summarize a broad spectrum of FNN optimization methodologies including conventional and metaheuristic approaches. This article also tries to connect various research directions emerged out of the FNN optimization practices, such as evolving neural network (NN), cooperative coevolution NN, complex-valued NN, deep learning, extreme learning machine, quantum NN, etc. Additionally, it provides interesting research challenges for future research to cope-up with the present information processing era.

1. Introduction

Back in 1943 McCulloch and Pitts (1943) proposed a computational model inspired by the human brain, which initiated the research on artificial neural network (ANN). ANNs are capable of learning and recognizing and can solve a broad range of complex problems. Feedforward neural networks (FNNs) are the special type of ANN models. The structural representation of an FNN makes it appealing because it allows perceiving a computational model (a function) in a structural/network form. Moreover, it is the structure of an FNN that makes it a universal function approximator, which has the capabilities of approximating any continuous function (Hornik, 1991). Therefore, a wide range of problems is solved by the FNNs, such as pattern recognition (Jain et al., 2000), clustering and classification (Zhang, 2000), function approximation (Selmic and Lewis, 2002), control (Lam and Leung, 2006), bioinformatics (Mitra and Hayashi, 2006), signal processing (Niranjan and Principe, 1997), speech processing (Gorin and Mammone, 1994), etc.

The structure of an FNN consists of several neurons (processing units) arranged in layer-by-layer basis and the neurons in a layer have connections (weights) from the neurons at its previous layer. Fundamentally, an FNN optimization/learning/training is met by searching an appropriate network structure (a function) and the weights (the parameters of the function) (Haykin, 2009). Finding a suitable network structure includes the determination of the appropriate neurons (i.e., activation functions), the number of neurons, and the arrangements of neurons, etc. Similarly, finding the weights indicates the optimization of a vector representing the weights of an FNN. Therefore, learning is an essential and distinguished aspect of the FNNs.

Numerous algorithms, techniques, and procedures were proposed in the past for the FNNs optimization. Earlier, in FNN research, only the gradient-based optimization techniques were the popular choices. However, gradually because of the limitations of gradient-based algorithms, the necessity of metaheuristic-based optimization methods were recognized.

Metaheuristics formulate the FNN components, such as weights, structure, nodes, etc., into an optimization problem. Metaheuristics implement various heuristics for finding a near-optimum solution. Additionally, a multiobjective metaheuristic deals with the multiple objectives simultaneously. The existence of multiple objectives in the FNNs optimization is evident since the minimization of FNN's approx-

* Corresponding author.

http://dx.doi.org/10.1016/j.engappai.2017.01.013
Received 7 December 2016; Received in revised form 20 January 2017; Accepted 23 January 2017
Available online 10 February 2017

imation error is desirable at one hand, and the generalization and model's simplification is at the other.

In a metaheuristic or multiobjective metaheuristic treatment to an FNN, an initial population of FNNs is guided towards a final population, where usually the best FNN is selected. However, selecting only the best FNN from a population may not always produce a general solution. Therefore, to achieve a general solution without any significant additional cost, an ensemble of many candidates chosen from a metaheuristic final population is recommended.

This article provides a comprehensive literature review to address the various aspects of the FNN optimization, such as:

1. The importance of an FNN as a function approximator and its preliminary concepts (Section 2), including the introduction to the factors influencing the FNN optimization (Section 2.2) and introduction to the conventional optimization algorithms (Section 2.3).
2. The role of metaheuristics and hybrid metaheuristics in FNNs optimization (Section 3).
3. The role of multiobjective metaheuristics (Section 4) and the ensemble methods (Section 5).
4. The current challenges and future research directions (Section 6).

2. Feedforward neural networks

The intelligence of human brain is due to its massively parallel neurons network system. In other words, the architecture of the brain. Similarly, a proper design of an ANN offers a significant improvement to a learning system. The components, such as nodes, weights, and layers are responsible for the developments of various ANN models.

A single layer perceptron (SLP) consists of an input and an output layer, and it is the simplest form of ANN model (Rosenblatt, 1958; Jain et al., 1996). However, SLPs are incapable of solving nonlinearly separable patterns (Minsky and Papert, 1988). Hence, a multilayer perceptron (MLP) was proposed, which addressed the limitations of SLPs by including one or more hidden layers in between an input and an output layer (Werbos, 1974). Initially, the backpropagation (BP) algorithm was used for the MLP training (Rumelhart et al., 1986). A trained MLP was then found capable of solving nonlinearly separable patterns (Rumelhart et al., 1986). In fact, MLPs (in general FNNs) are capable of addressing a large class of problem pertaining to pattern recognition and prediction. Moreover, an FNN is considered as a **universal approximator** (Hornik, 1991). Cybenko (1989) referring to Kolmogorov's theorem[1] showed that an FNN with only a single internal hidden layer—containing a finite number of neurons with any continuous sigmoidal nonlinear activation function—can approximate any continuous function. Also, the FNN structure (architecture) is itself capable enough to be a universal approximator (Hornik et al., 1989; Hornik, 1991). Hence, several researchers praised FNN for its universal approximation ability (Kůrková, 1992; Leshno et al., 1993; Huang and Babri, 1998; Huang et al., 2006a).

Many other ANN models, like radial basis function (Lowe and Broomhead, 1988) and support vector machine (Cortes and Vapnik, 1995) are a special class of three-layer FNNs. They are capable of solving regression and classification problems using supervised learning methods. In contrast, adaptive resonance theory (Grossberg, 1987), Kohenen's self-organizing map (Kohonen, 1982), and learning-vector-quantization (Kohonen, 1982) are two-layer FNNs that are capable of solving pattern recognition and data compression problems using unsupervised learning methods.

Additionally, the ANN architecture with feedback connections, in other words, a network where connections between the nodes may form cycles is known as a **recurrent neural network** (RNN) or feedback network model. The RNNs are good at performing sequence recognition/reproduction or temporal association/prediction tasks. RNNs such as Hopfield network (Hopfield, 1982) and Boltzmann machine (Ackley et al., 1985) are good at the application for memory storage and remembering input–output relations. Moreover, Hopfield network was designed for solving nonlinear dynamic systems, where the stability of a dynamic system is studied under the neurodynamic paradigm (Hopfield, 1982).

A collection of RNN models, such as temporal RNN (Dominey, 1995), echo state RNN (Jaeger, 2001), liquid state machine (Natschläger et al., 2002) and backpropagation de-correlation (Steil, 2004) forms a paradigm called reservoir computing, which addresses several engineering applications including nonlinear signal processing and control. Although some other ANN models that are capable of doing a similar task that of the FNNs were pointed out in this Section, the discussion in this article is; however, limited to only FNNs.

2.1. Components of FNNs

FNNs are the computational models that consist of many neurons (*node*), which are connected using synaptic links (*weights*) and are arranged in layer-by-layer basis. Thus, the FNNs have a specific structural configuration (*architecture*) in which the nodes at a layer have forward connections from the nodes at its previous layer (Fig. 1(a)). A node of an FNN is capable of processing information coming through the connection weights (Fig. 1(b)). Mathematically, the output y_i (excitation) of a node (node indicated as i) is computed as:

$$y_i = \varphi_i\left(\sum_{j=1}^{n^i} w_j^i z_j^i + b^i\right), \tag{1}$$

where n^i is the total incoming connections, z^i is the input, w^i is the weight, b^i is the bias, and $\varphi_i(\cdot)$ is the *activation function* at the i-th node to limits the amplitude of the output the node into a certain range.

Fig. 1(a) is a structural representation of an FNN, i.e., a phenotype of a function $f(\mathbf{x}, \mathbf{w})$, which is parameterized by a p-dimensional input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_p\rangle$ and an n-dimensional real-valued weight vector $\mathbf{w} = \langle w_1, w_2, \ldots, w_n\rangle$. The function $f(\mathbf{x}, \mathbf{w})$ is a solution of a problem. Therefore, two tasks involved in solving a problem using an FNN are: to discover an appropriate function $f(\mathbf{x}, \mathbf{w})$ (i.e., the architecture optimization) and to discover an appropriate weight vector $\mathbf{w}$ (i.e., the weights optimization) using some *learning algorithm*.

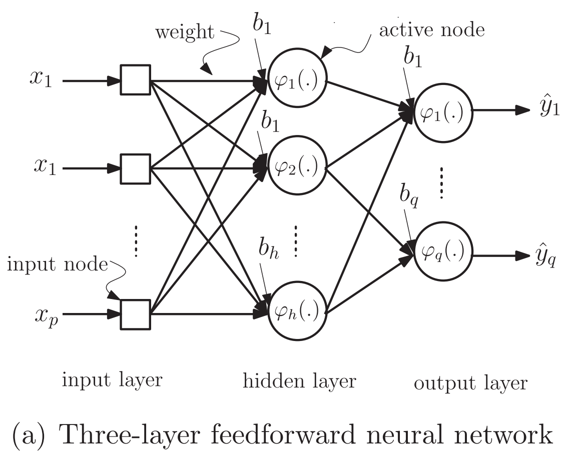
The architecture optimization indicates the search for the appropriate activation functions at the nodes, the number of nodes, number of layers, the arrangements of the nodes, etc. Therefore, several components of an FNN optimization are: the connection **weights**; the **architecture** (number of layers in a network, the number of nodes at the hidden layers, the arrangement of the connections between nodes); the **nodes** (activation functions at the nodes); the **learning algorithms** (algorithms training parameters); and the **learning environment**. However, traditionally, the only component that was optimized was the weights of the connections by keeping other components fixed to the initial choice.

2.2. Influencing factors in FNN optimization

2.2.1. Learning environments

An FNN is trained by supplying the training data (X, Y) of N input–output pairs, i.e., $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ and $Y = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$. Each input $\mathbf{x}_i = \langle x_{i1}, x_{i2}, \ldots, x_{ip}\rangle$ is a p-dimensional vector, and it has a corresponding q-dimensional desired output vector $\mathbf{y}_i = \langle y_{i1}, y_{i2}, \ldots, y_{iq}\rangle$. For the training data (X, Y), an FNN produces an output $\hat{Y} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_N)$, where a vector $\hat{\mathbf{y}}_i = \langle \hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{iq}\rangle$ is a q-dimensional FNNs output, which is then compared with the desired output $\mathbf{y}_i$, for all $i = 1$ to N by

[1] Kolmogorov's theorem: "All continuous functions of n variables have an exact representation in terms of finite superpositions and compositions of a small number of functions of one variable (Kolmogorov, 1957)."



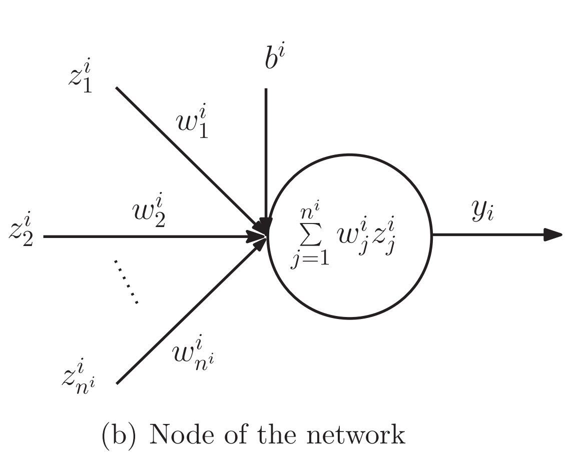


Fig. 1. Three-layer feedforward neural network (a), where input layer has p input nodes, hidden layer has h activation functions, and output layer has q nodes.

using some error/distance/cost function. The minimization/reduction of the error/distance function, in an iterative manner, is referred as a **supervised learning**. One very commonly known supervised learning algorithm is Delta rule or Widrow-Hoff rule (Wisrow et al., 1960; Widrow, 1959) in which the n-dimensional weight vector $\mathbf{w}$ of an FNN is optimized as:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \Delta\mathbf{w}^t, \tag{2}$$

where Δw^t is weight change (an additive term) at t-th iteration. The weight change Δw^t is computed as:

$$\Delta\mathbf{w}_i^t = \eta^t e_i^t \mathbf{x}_i^t, \tag{3}$$

where η^t is a learning rate, which controls the magnitude of weight change at t-th iteration and e_i^t is the error at t-th learning iteration corresponding to i-th training input $\mathbf{x}_i^t$ presented to an FNN. The error $e^t{}_i$ at the t-th iteration may be computed as: $e_i^t = \sum_{j=1}^q (y_{ij}^t - \hat{y}_{ij}^t)^2$, where y_{ij}^t and $\hat{y}_{ij}^t$ are the desired output and FNN's output at t-th iteration respectively.

Contrary to the supervised learning paradigm, there are two other learning forms for the spacial cases of FNNs: (1) the *unsupervised learning*—for the unlabeled training data (Rumelhart and Zipser, 1985), and (2) the *reinforcement learning*—for the training data with insufficient input–output relations (Kaelbling et al., 1996). The focus of this article is, however, on supervised learning paradigms only.

2.2.2. Error functions

A supervised learning, essentially, is the minimization of the difference/distance between the desired output $\mathbf{y}_i$ and the model's output $\hat{\mathbf{y}}_i = f(\mathbf{x}, \mathbf{w})$ by comparing the difference/distance using a cost function $c_f: Y \times \hat{Y} \longrightarrow \mathbb{R}_{\geq 0}$. For this propose, several cost function can be designed. For instance, in regression problems, *mean squared error* is one of the commonly used cost function, which is written as:

$$c_f(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{q}\left(y_{ij} - \hat{y}_{ij}\right)^2, \tag{4}$$

where y_{ij} are the desired response and $\hat{y}_{ij}$ are the FNN's responses, and their differences were summed over N data pairs. Some other functions like sum of squared error, root of mean square error, mean absolute error, correlation coefficient, etc., can be used for evaluating the FNN's predictability (Pearce and Ferrier, 2000).

The cost function (4) or any similar squared-error-based cost function is inconsistent for solving classification problems (Twomey and Smith, 1995). Instead, the *percentage of good classification*, which has consistent behavior, can be used (Twomey and Smith, 1995). However, the percentage of good classification is satisfactory until no preference was given to a particular class. Therefore, *accuracy* and *miss-classification rate* are used as the cost functions. A detailed list of the cost function for evaluating the classification problems is provided by Pencina et al. (2008), Sokolova and Lapalme (2009), and FernandezCaballero et al. (2010).

In this article, cost function mentioned for FNN optimization is discussed in a general sense, which can be thought as the equivalent to any other user-defined cost function. Another factor related to fitness of an FNN is to compare the cost functions of two or more FNN models (Baranyi et al., 1999; van der Voet, 1994). Some researchers also argue to statistically compare the predicted outputs of two or more FNN models to establish the differences in their performances (Diebold and Mariano, 1995).

2.2.3. Local minima problem

Let $c_f: S \longrightarrow \mathbb{R}_{\geq 0}$, where $S \subset \mathbb{R}^n$ is nonempty and compact (for detailed information about topological compactness, see Simovici and Djeraba (2008)). Therefore, the following may be defined:

Definition 1. A point $\mathbf{w}^* \in S$ is called **global minima** if $c_f(\mathbf{w}^*) \leq c_f(\mathbf{w})$ for any $\mathbf{w} \in S$ holds.

Definition 2. A point $\mathbf{w}^* \in S$ is called **local minima** if there exists $\epsilon > 0$, and an ϵ-neighborhood $B_\epsilon(\mathbf{w}^*, \epsilon)$ around $\mathbf{w}^*$ such that $c_f(\mathbf{w}^*) \leq c_f(\mathbf{w})$ for any $\mathbf{w} \in S \cap B_\epsilon(\mathbf{w}^*, \epsilon)$ holds.Learning algorithms when to using the cost function (4) or any similar function for FNN optimization has the tendency to fall in local minima (Hansen and Salamon, 1990). Moreover, the geometrical structure (parameter space) of a three-layer perceptron may fall to local minima and plateaus during its optimization. It indicates that the critical point corresponding to global minima of a smaller FNN model (model with $h-1$ hidden units) can be a local or saddle point of a larger FNN model (model with h hidden units) (Fukumizu and Amari, 2000). However, there are some ways to avoid or eliminate local minima in FNN optimization (Wessels and Barnard, 1992; Toh, 2003):

(1) If the weights and training patterns are assigned randomly to a three-layer FNN that contains h neurons at the hidden layer, then a gradient-descent algorithm can avoid trapping into local minima (Poston et al., 1991).
(2) If linearly-separable training data and pyramidal network structure are taken, then the error surface will be local minima free (Gori and Tesi, 1992).
(3) If there are N many noncoincident input patterns to learn and three-layered FNN with $N-1$ sigmoid hidden neurons and one dummy hidden neuron is used, then the corresponding error surface will be local minima free.
(4) If the training algorithms can be improved as similar to as the global descent learning algorithm proposed by Cetin et al. (1993) to replace gradient-descent algorithms, then it can avoid local minima.

These four methods depend on the number of hidden neurons, the number of training samples, the number of output neurons, and a condition that says the number of hidden neurons should not be less than the number of training samples. Moreover, it does not necessarily guarantee to converge to a global minima and to set the preconditions for the number of hidden neurons and linearly separable training patterns are unlikely conditions for the real-world problems (Huang, 1998).

2.2.4. Generalization

The generalization is a crucial aspect of an FNN optimization, where it is an ability to offer the general solutions rather than performing best for the particular cases. To achieve generalization, FNNs need to avoid both *underfitting* and *overfitting* during training, which is associated with high statistical *bias* and high statistical *variance* (Geman et al., 1992). Therefore, one has to address trade-offs between bias and variance. Also, for a good generalization, the number of training pattern should be sufficiently larger than the total number of connections in FNN (Widrow and Lehr, 1990).

The standard methods to achieving generalization are determining an optimum number of free parameters (i.e., equivalent to find an optimum network architecture), *early stopping* of training algorithms, *adding regularization term* with the cost function (Girosi et al., 1995; Bishop, 1995), and *adding noise* to the training data.

In *early stopping*, a dataset is divided into three sets: a training set, cross-validation set, and test set. The early stopping scheme suggests stopping of training at the point (epoch) from which onward the cost function value computed on cross-validation set starts to rise (Hansen and Salamon, 1990; Amari et al., 1997; Prechelt, 1998; Yao et al., 2007). Similarly, adding noise (jitters) into the training pattern improves FNN's *generalization* ability and removing insignificant weights from a trained FNN improves its *fault tolerance* ability (Murray and Edwards, 1994). Moreover, generalization is related to sparsity and stability of a learning algorithm (Bousquet and Elisseeff, 2002).

Now, if the approximation error of two FNN models trained on the same training data is close/similar, then the model with simple network structure (lower number of free parameters) should be selected as the best model. It is because the model with lower network complexity possesses higher generalization ability than the models with higher network complexity (Reed et al., 1995). Moreover, the network with lower weight magnitude possesses better generalization ability (Reed et al., 1995).

2.3. Conventional optimization approaches

Finding a suitable algorithm for the FNNs optimization has always been a difficult task. The FNN optimization using conventional gradient based algorithms is viewed as an unconstrained optimization problem (Haykin, 2009; Lippmann, 1987). The cost function c_f has to be optimized to satisfy Definition 1. Therefore, the gradient of error g^t at t-th iteration is computed as:

$$g^t = \frac{\partial c_f}{\partial \mathbf{w}^t}, \tag{5}$$

where g^t is a *first-order partial derivative* of the cost function c_f with respect to weight vector $\mathbf{w}$. Hence, a gradient-descent approach starts with an initial guess $\mathbf{w}_0$ and generates a sequence of weight vector $\mathbf{w}_1, \mathbf{w}_2, \ldots$ such that c_f reduces in each iteration. The connection weights at iteration t are updated as:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \Delta\mathbf{w}^t, \tag{6}$$

where the weight change $\Delta\mathbf{w}^t$ is equal to $-\eta^t g^t$, and η^t is the learning rate. The weights updated using (5) and (6) is known as the steepest decent approach. Now, instead of using a first-order partial derivative, a *second-order partial derivative* (∇^2) of cost function c_f can be used as:

$$H^t = \nabla^2 c_f = \frac{\partial^2 c_f}{\partial \mathbf{w}}, \tag{7}$$

where H^t is *Hessian* matrix at the t-th iteration (Chen and Sheu, 1994). Hence, the weight change $\Delta\mathbf{w}^t$ using second–order Taylor's series expansion of cost function c_f around point $\Delta w^t = H^{t-1} g^t$ is computed as:

$$\Delta\mathbf{w}^t = -H^{t^{-1}} g^t, \tag{8}$$

where $H^{t^{-1}}$ is the inverse of Hessian matrix H^t and the weight change $\Delta\mathbf{w}^t$ is known as the *Newton method* or Newton update (Haykin et al., 2009). In the past, several algorithms were proposed using (5) and (8). Some of them are summarized as follows:

Backpropagation (BP) is a **first-order gradient-descent** algorithm for the FNNs optimization (Werbos, 1974; Rumelhart et al., 1986). In BP, the error computed at the output layer is propagated backward to the hidden layers. BP algorithm has two phases of computation: *forward computation* and *backward computation*, where at t-th iteration, the weight change $\Delta\mathbf{w}^t$ for l-th layer is computed as:

$$\Delta\mathbf{w}_l^t = \alpha^t \mathbf{w}_l^{t-1} + \eta^t g^t \mathbf{y}_{l-1}, \tag{9}$$

where $\mathbf{y}$ is inputs/excitation from previous layer $l-1$, η^t is learning rate and α^t is momentum factor.

The choice of learning rate η^t and momentum factor α^t are critical to gradient-descent technique. The momentum factor α^t allows BP training to be biased with previous iteration weights that help convergence rate to be faster. BP is sensitive to these parameters (Rumelhart et al., 1986). If the learning rate is too small, learning will become slow, and if the learning rate is too large, learning will be zigzag and algorithm may not converge to required degree of satisfaction. Additionally, a high momentum factor leads to a high risk of overshooting minima and a low momentum factor may avoid local minima, but learning will be slow. The classical BP algorithm is slow and has a tendency to fall in local minima (Gori and Tesi, 1992).

Since the basic version of BP is sensitivity towards learning rate and momentum factor (Rumelhart et al., 1986), several improvements were suggested by researchers: (1) a fast BP algorithm, called *Quickpro* was proposed by Fahlman and Lebiere (1990), Fahlman (1988); (2) a *delta-bar* technique and an *acceleration* technique was suggested for tuning BP learning rate η by Jacobs (1988) and Silva and Almeida (1990) respectively; and (3) a variant of BP, called resilient propagator (*Rprop*) was proposed by Riedmiller and Braun (1993).

In the *Rprop*, if the gradient direction in iteration n remains unchanged from its previous iteration $t-1$, then the weight change will occur in larger magnitude, else in smaller. In simple words, if gradient sign remains unchanged from previous iterations, the magnitude of learning rate η will be large, otherwise small. The proposed *Rprop* improves determinism of convergence to global minima (Riedmiller and Braun, 1993). However, it is not faster than the *Quickpro*, but still faster than BP (Schiffmann et al., 1994).

Contrary to BP, a **second-order minimization method**, called *conjugate gradient* (CG) can be used for weights optimization (Hestenes and Stiefel, 1952; Barnard and Cole, 1989; Charalambous, 1992). The CG does not proceed down with a gradient; instead, it moves in the direction that is conjugate to the direction of the previous step. In other words, the gradient corresponding to the current step stays perpendicular to the direction of all the previous steps, and each step is at least as good as its previous step. Such series of steps are non-interfering. Hence, the minimization performed in one step will not be undone by any further steps. Several variants of the CG were proposed in the past (Dai and Yuan, 1999).

Similar to the CG, many other variants of derivative-based conventional methods are used for weights optimization: *Quasi-Newton* (Chen and Sheu, 1994), *Gauss-Newton* (Bertsekas, 1999), or

Levenberg-Marquardt (Marquardt, 1963). Quasi-Newton uses a second-order partial derivative (7) of error (4), and it computes its weight search direction by using Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Fletcher, 1987). In Gauss-Newton method, the FNNs optimization is framed as a nonlinear least square optimization problem, which suggests to using the sum of squared error (4) (Marquardt, 1963). Many researchers suggested that the Levenberg-Marquardt (LM) method outperforms BP, CG, and Quasi-Newton methods (Hagan and Menhaj, 1994; Lera and Pinzolas, 2002). Several other methods were proposed for the FNNs optimization are based on *Kalman-filter* (Haykin et al., 2001; Sum et al., 1999) and *recursive least squares method* (Azimi-Sadjadi and Liou, 1992).

2.4. Comments on conventional approaches

The gradient-descent based conventional algorithms operate on a single solution (a weight vector) during the optimization. Thus, these algorithms are computationally faster than the algorithms that use two or more solution vectors during the optimization and select the best solution vector at the end of optimization iterations. Moreover, the gradient-decent methods such as BP (Rumelhart et al., 1986), Online BFGS (Schraudolph et al., 2007) can be applied for the stochastic as well as batch mode training of the FNNs.

The basic advantages of the stochastic/online training of an FNN are its ability to address redundancy in training pattern, the inclusion of training data that are currently not in training set (i.e., the possibility of dynamic learning), and faster training than that of batch mode (Wilson and Martinez, 2003). Whereas, most of the other second-order gradient-descent methods and metaheuristic algorithms can only use batch mode training. However, a batch mode (offline) training of an FNN can at least guarantee a local minima under a simple condition compared to a stochastic/online training, and for a larger dataset, batch mode training can be faster than stochastic training (Nakama, 2009).

On the contrary to the advantages, the conventional methods have several limitations. For example, they have the tendency to fall in local minima, and they are only used for optimizing the FNN weights. Additionally, the gradient-descent algorithms depend on the error function, e.g., mean square error or sum of squared error. For instance, the least square methods like the Gauss-Newton and LM works only if the cost function is the sum of squared error. The Newton method has to compute a Hessian matrix (7), which has to be positive definite and computing the Hessian matrix (7) can be hard and expensive. Similarly, the Quasi-Newton and CG methods need to use a line-search method that sometimes can be expensive.

Moreover, the FNN generalization, as mentioned in Section 2.2.4, needs the reduction in the number of weights (less complex network architecture). Hence, the application of conventional algorithms is limited compared to the metaheuristic algorithms such as the genetic algorithm (GA) that can be directly applied to an FNN for its automatic structure determination and complexity reduction (Holland, 1975; Goldberg and Holland, 1988). Similarly, a metaheuristic algorithm can formulate an FNN such that the insignificant weights of the network can be eliminated to improve the FNN generalization ability. Moreover, metaheuristic algorithms can evolve an FNN as a whole by optimizing its components simultaneously.

3. Metaheuristic approaches

So far, only the gradient-descent based algorithms were discussed, which are local search algorithms. They are good at exploiting the obtained solutions to find new solutions. However, to find a global optimum solution, any optimization algorithm must use two techniques: (1) *exploration*—to search new and unknown areas in a search space and (2) *exploitation*—to take advantage of the already discovered solution (March, 1991). The exploration and exploitation are two contradictory strategies and a good search algorithm must find a trade-off between these two. Metaheuristic is the procedure that implements nature-inspired heuristics to combine these two strategies (Osman and Laporte, 1996). Hence, metaheuristic approaches are alternative to the conventional approaches for optimizing the FNNs.

Unlike the conventional methods, which require the objective function to be continuous and differential, the metaheuristic algorithms have the ability to address complex, nonlinear, and non-differentiable problems. However, the optimization algorithms are often biased towards a specific class of problems, that is, "there is no such universal optimizer which may solve all class of problem," which is evident from no free lunch theorem (Wolpert and Macready, 1997).

Wolpert and Macready (1997) introduced *no free lunch* (NFL) theorem to answer the question, "whether a general purpose optimization algorithm exists." Moreover, Wolpert (1996) introduced NFL for optimization algorithm to answer the question, "How does the set of problems $F_1 \subset \mathcal{F}$ for which algorithm a_1 performs better than algorithm a_2 compares to the set $F_2 \subset \mathcal{F}$ for which the reverse is true." Here, $\mathcal{F}$ is space of all possible problems. To answer this question, Wolpert proposed NFL theorem, which says that "the average performance of any pair of algorithms across all possible problems is identical" (Wolpert and Macready, 1997).

Therefore, a straightforward interpretation of NFL is as follows. "A general purpose universal optimization strategy is impossible, and the only way one strategy can outperform another if it is specialized to the structure of the specific problem under consideration" (Ho and Pepyne, 2002). Schumacher et al. (2001) argue that the NFL theorem (Wolpert and Macready, 1997) holds only for the set of problems which are closed under permutation (c.u.p). Therefore, indeed the performance of one algorithm can be improved over another for the problems that are not c.u.p and most of the real-world problems are not c.u.p (Igel and Toussaint, 2003). Such is the reason why in the past researchers were inclined to create and improvise algorithms for optimizing the FNNs.

3.1. Metaheuristic algorithms

Since a large variety of metaheuristic algorithms are available, it is difficult to classify metaheuristic algorithms precisely into different classes. Though, intuitively, three basic categorize can be done:

3.1.1. Single solution based algorithms

A single solution based metaheuristic algorithm operates on a single solution (candidate) and applies some heuristic inspired by the nature or some other phenomena on the current solution. For example, algorithms like simulated annealing (SA) (Kirkpatrick et al., 1983), tabu search (TS) (Glover, 1989), variable neighborhood search (VNS) (Mladenović and Hansen, 1997), greedy randomized adaptive search (GRAP) (Feo and Resende, 1995), etc., improves a single solution by searching around its neighborhood and continue to improve the solution until a satisfactory solution is obtained. For instance, the heuristics of some algorithms are as follows:

SA is a probabilistic approach that imitates the cooling strategy (annealing process) of a metallurgy industry. It uses Monte Carlo method (Metropolis et al., 1953) to determine the acceptance probability of a newly generated solution in the neighborhood of the current solution. Hence, for a given search space, SA should guide a solution towards a global optimal solution (Kirkpatrick et al., 1983; Černỳ, 1985). Similarly, TS is inspired by the human behavior of tabooing objects (Glover, 1989). In other words, TS discourages (tabu) the acceptance of the solutions that are already explored in the past. Hence, it improves upon SA by introducing some additional restriction on the acceptance of the new solutions.

Since a single solution based algorithm exploits the current solution, it also is known as the *local search algorithm*. The focus of this article is to illustrate the application of the metaheuristic algorithms for the FNNs optimization. Hence, for the detailed description of the mentioned algorithms, the readers may explore the respective references.

3.1.2. Population based algorithms

Population based algorithms operate on the multiple solutions (candidates) and apply the heuristics inspired by nature, biological evolution, biology, or some other forms. In contrast to the single solution based algorithms, the population based algorithms have a high exploration (global search) ability. The following are the population based algorithms:

Evolutionary algorithms (*EA*). Genetic algorithms (GA) (Holland, 1975; Goldberg and Holland, 1988), evolutionary programming (EP) (Fogel, 1998), evolutionary strategy (ES) (Schwefel, 1987), genetic programming (GP) (Koza et al., 1999), differential evolution (Storn and Price, 1997), etc., are the algorithms inspired by the dynamics of natural selection and use the operators, such as *selection*, *crossover*, and *mutation* to find a near-optimal solution (Goldberg and Holland, 1988). EA framework offers an exploration of a vast search space and guarantees to find a near-optimal solution. Since EAs do not depend on gradient information, they solve a large range of complex, nonlinear, nondifferentiable, multimodal optimization problems. Also, EAs give a wider scope in the FNN optimization since EAs can optimize both discreet and continuous optimization problem, and the FNN components can be formulated into both ways.

The differences between EAs can be briefly stated as follows: GA uses genetic operators, such as selection, crossover, and mutation to search optimum genetic vector from a search space (Goldberg and Holland, 1988); whereas, only the mutation operator are used in ES to evolve a real vector solution (Schwefel, 1987). On the other hand, GP searches an optimum program structure from a topological search space of computer programs (Koza et al., 1999) and EP are used for evolving parameters of a computer program whose structure is kept fixed (Fogel, 1998).

Swarm intelligence (*SI*). SI algorithms are inspired by the collective and self-organized behavior of the swarm (insects, birds, fish, etc.). Particle swarm algorithm (PSO) (Eberhart and Kennedy, 1995), ant colony optimization (ACO) (Dorigo et al., 1996), artificial bee colony (ABC) (Karaboga, 2005), bacterial foraging optimization (BFO) (Passino, 2002), etc., are some widely used SI algorithms. The basic principle of SI algorithms is as follows. First, a swarm (collection of solutions) are randomly generated. Then, the heuristic inspired by the swarm behavior modifies the current solution. For example, in PSO, ACO, ABC, and BFO, the heuristics are inspired by the foraging behavior of birds, ants, bees, and bacteria respectively. In these algorithms, an FNN component is formulated as a solution for the optimization.

In PSO, a swarm, as a whole, is like a flock of birds (particles, which are the weight vectors) that collectively foraging (explore search space) for food (best weight vector) and is likely to move close to an optimum food-source (Eberhart and Kennedy, 1995; Kennedy et al., 2001). Moreover, each particle bears two properties: location and velocity. The location of a particle is a solution vector (weight vector $\mathbf{w}$), and velocity η is a vector equal to the size of the location vector. Each particle determines its movement using knowledge of its best locations, global best location, and random perturbations (Eberhart and Kennedy, 1995; Shi and Eberhart, 1998).

In ACO, the artificial ants explore the area around their nest (colony) for searching a food source. ACO takes advantage of ants ability to choose the shortest path to a food source by communicating among each other's using a pheromone secretion (Deneubourg et al., 1990). This behavior of ants led to the development of ACO algorithm (Dorigo et al., 1996).

Similarly, in ABC, three kinds of honey bees, such as employed bee, onlooker bee, and scout bee are responsible for searching food source (Karaboga, 2005). Each employed bee memorizes a food source, i.e., a solution (weight vector). Then, each onlooker bee examines the nectar amount (fitness of solution) of a food source memorized by the employed bees and depending on nectar amount; they send scout bees for searching new food source. Hence, they iteratively construct the solution. The readers are encouraged to explore the detail description the algorithms in their respective references.

Other metaheuristics. The population based metaheuristic algorithms metaphor is exploited to device several algorithms. There are algorithms inspired by the behavior of animals, birds, and insects, such as gray wolf optimization (GWO) (Mirjalili et al., 2014), flower pollination (FP) (Yang, 2012), cuckoo search (CS) (Yang and Deb, 2009), firefly (FF) (Yang, 2010), etc.

Similarly, there are algorithms inspired by some phenomenon observed in the physics and chemistry, such as harmony search (HS) (Geem et al., 2001), central force optimization (CFO) (Formato, 2007), gravitational search optimization (GSO) (Rashedi et al., 2009), etc. A detailed list and classification of metaheuristic algorithms are provided by Fister et al. (2013).

The growing number of metaheuristic algorithms has drawn researchers to examine the metaphor, the novelty, and the significant differences among the metaheuristic algorithms (Weyland, 2010; Sörensen, 2015). Sörensen (2015) provided an insight of the metaheuristic developed over the time, starting from SA to TS, EA, ACO, HS, etc., where the author claimed that most of the metaheuristic are similar in nature and do not provide a groundbreaking method in optimization. Despite the criticisms, the author acknowledged the quality of metaheuristic research has been produced and can be produced.

3.1.3. Hybrid and memetic algorithms

An effective combination of various metaheuristic algorithms may offer a better solution than that of a single algorithm. A paradigm of hybrid algorithms, called *memetic algorithm* gave a methodological concept for combining two or more metaheuristics (global and local) to explore a search space efficiently and to find a global optimal solution (Moscato, 1989).

The conventional algorithms have the local minima problem because they lack global search ability, but they are fast and good in local search. On the other hand, the metaheuristics are good in global search, but they suffer premature convergence (Leung et al., 1997; Trelea, 2003). Therefore, a combination of these two may offer a better solution in FNN optimization than that of using any one of them alone (Fig. 2). To reach a global optimum, a hybrid strategy can be used. Fig. 2 shows an impact of hybridization of metaheuristics on the FNNs optimization. The hybrid algorithms can be categorized in two paradigms:

(1) The combination of conventional and metaheuristic algorithms—to take advantage of local search and global search algorithms.

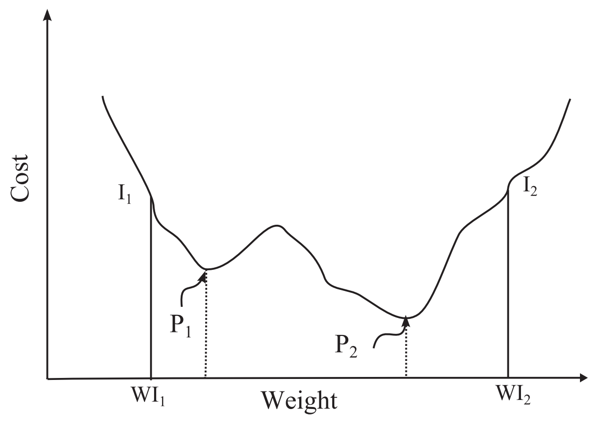


Fig. 2. Metaheuristic may be used for finding initial weights WI$_2$ and conventional algorithms may be for finding global optima P$_2$ and vice versa (Yao, 1993).

(2) The combination of two or more metaheuristic algorithms—to make use of different heuristics or a combined influence of two or more heuristics of global search algorithms.

Under the definition of memetic algorithms, researchers combine EAs with conventional algorithms (Brownlee, 2011). For example, the effectiveness of global (GA) and local search (BP) combination is explained by Yin et al. (2011) and Yaghini et al. (2013). Similarly, a hybrid PSO and BP algorithms for optimizing FNN were found useful in obtaining better approximation than using one of them alone (Zhang et al., 2007a).

A convergence scenario similar to Fig. 2 was illustrated in Ozturk and Karaboga (2011), where ABC was applied for searching initial weights and LM was applied for optimizing the already discovered weights. An example of effectively combining two metaheuristic GA and PSO is illustrated by Juang (2004), where both GA and PSO optimized the same population. A detailed description of the hybrid metaheuristic algorithms for the FNN optimization is described in the following Section.

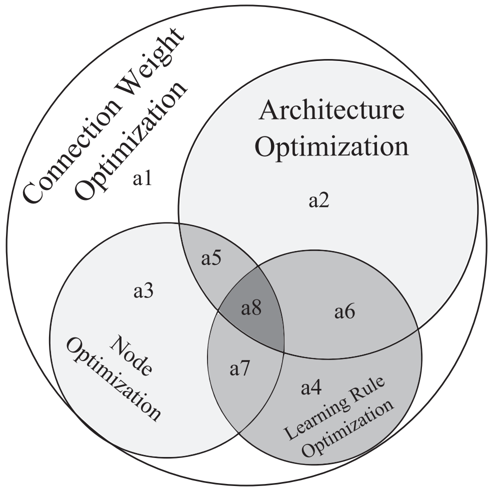
3.2. Metaheuristic formulation of the FNN components

Metaheuristics are stochastic/non-deterministic algorithms. Hence, they do not guarantee a global optimal solution, but they can offer a near-optimal (satisfactory) solution. Moreover, metaheuristics efficiently solve a wide range of complex continuous optimization problems; especially when the problems have incomplete and imprecise information.

The basic form of FNN optimization is the act of searching its weights (free parameters of FNN) such that the cost function (4) or similar function can be minimized. However, the goodness (performance) of FNN cost function depends not only on finding optimum weights, but finding the optimum architecture, activation function, parameter setting of learning algorithm, and training environment are equally important. To apply metaheuristic algorithms for optimizing an FNN, its components (*phenotype*) need to be formulated into a vector (*genotype*) form.

Usually, the FNN components, such as weights, architecture, activation function, learning rule, etc., are considered arbitrarily. Then, a learning algorithm is applied to search weights while the other components are kept fixed to their initial setting. The metaheuristics, on the other hand, allow us to optimize each component simultaneously or a combination of components efficiently (Fig. 3).

The *Venn diagram* in Fig. 3 illustrates the spectrum of FNN components optimization: weights, architecture, activation function, and learning rule's parameters. In Fig. 3, area "a1" indicates the optimization of weights; area "a2" indicates the optimization of weights and architecture; area "a3" indicates the optimization of weights, architecture, and activation function; and all other possible combinations. Examining Fig. 3, one can say that the strength and complexity of optimization increases from area denoted "a1" to "a8," where "a1" is the simplest approach and "a8" is the most sophisticated approach.



Connection Weight Optimization
Architecture Optimization
a1
a2
a3
a4
a5
a6
a7
a8
Node Optimization
Learning Rule Optimization

Fig. 3. Spectrum of metaheuristic design of FNN.

Each FNN component can be separately optimized on a one-by-one basis. Therefore, firstly, the weights can be optimized by keeping a fixed architecture; secondly, the architecture can be optimized keeping weights fixed; thirdly, the activation function can be optimized keeping architecture and/or weights fixed; and so on. Another way is to optimize all or a combination of FNN components simultaneously. Therefore, weights and architecture can be optimized, simultaneously; or weights and activation functions, simultaneously; or weights, architecture, and activation functions simultaneously; and so on. In the simultaneous optimization of all or a combination of components, a vectored representation of all components, or a combination of components can be optimized respectively. Once a vectored representation (genotype) is designed, then one of the available metaheuristic algorithms in the literature can be applied to optimize the designed vector to obtained an optimum FNN.

Now, there are *single-solution* based and *population* based metaheuristics (Boussaid et al., 2013). In a single-solution based metaheuristic algorithm, a genotype $\mathbf{w} = \langle w_1, w_2, \ldots, w_n \rangle$ is used. Whereas, in a population-based metaheuristic algorithm, a collection of many genotypes are used. In other words, a population matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$ of m weight vectors is used.

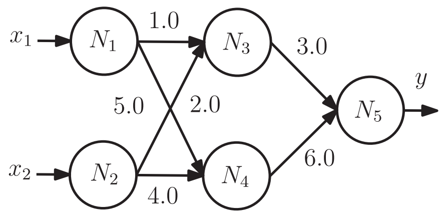
Yao and Liu (1998a) identified evolution at various components of FNN that fall into the spectrum of metaheuristic design of FNN shown in Fig. 3. This Section will describe *how researchers applied metaheuristics for evolving FNN*. The evolution in FNN components is described here one-by-one, as follows. Here, the word optimization, adaptation, and evolution are used in the similar context.

3.2.1. Weight optimization

FNN weight optimization is the most common and widely studied approach, in which the weights are mapped onto an n-dimensional weight vector $\mathbf{w}$, where n is the total number of weights in a network. The vector $\mathbf{w}$ is a genotype representation of a phenotype (FNN structure), where the weight $\mathbf{w} \in \mathbb{R}^n$. The weights w_i, an element of vector $\mathbf{w}$, may be encoded in the following ways: by assigning a real value, l-bits binary coding, l-bits gray coding, IEEE floating point coding, etc. Fig. 4 is an example of phenotype to genotype mapping, where a phenotype shown in Fig. 4(a) that has the connectivity matrix c as per Fig. 4(b) is encoded into three different weight vectors shown in Fig. 4(c).

FNN weights optimization using metaheuristic is practiced from **early 80's** when even the term metaheuristic was not used. Engel's (1988) work on FNN weight vector optimization using SA was the first evidence of metaheuristic application. To optimize weight vector using SA, first, the phenotype was mapped onto a real-valued weight vector (Fig. 4(c)), and to compute fitness of the FNN, a *reverse mapping* from genotype (weight vector $\mathbf{w}$) to phenotype (FNN) was used. Such process was continued until a satisfactory solution was found. SA based FNN weight optimization was found to be performing better in comparison to conventional approaches (Shang and Wah, 1996; Sexton et al., 1999; Sarkar and Modak, 2003).

Similar to Engel's (1988) approach of phenotype to genotype mapping and vice versa, Beyer and Ogier (1991) performed the FNN weights optimization using TS. Battiti and Tecchiolli (1995) used an improved TS, called reactive tabu search for optimizing weights. Several studies show that TS when used for optimizing FNN weights, outperformed BP and SA algorithm (Sexton et al., 1998a; Ye et al., 2007). However, SA and TS are single solution based algorithms, which has a limited scope of exploring search space to obtain a global optimal solution. In contrast, the EAs, SI, or other bio-inspired metaheuristics are population-based algorithms that operate on multiple agents to



(a)

$$c = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \mathbf{0} & 0 & 1 & 1 & 0 \\ 2 & 0 & \mathbf{0} & 1 & 1 & 0 \\ 3 & 0 & 0 & \mathbf{0} & 0 & 1 \\ 4 & 0 & 0 & 0 & \mathbf{0} & 1 \\ 5 & 0 & 0 & 0 & 0 & \mathbf{0} \end{array}$$

(b)

(i) $\mathbf{w} = \langle 1.0, 5.0, 2.0, 4.0, 3.0, 6.0 \rangle$

(ii) $\mathbf{w} = \langle 0001, 0101, 0010, 0100, 0011, 0110 \rangle$

(iii) $\mathbf{w} = \langle 0001, 0111, 0011, 0110, 0010, 0101 \rangle$

(c)

Fig. 4. Mapping of phenotype to genotype. (a) Phenotype of a three-layer FNN. (b) Adjacency matrix. (c) Weights encoding: (i) real value; (ii) 4-bits binary; (iii) 4-bits gray encoding.

explore a search space. Hence, they have a better exploration ability than SA, TS, BP, CG, and other single solution based algorithms (Goldberg and Holland, 1988; Kennedy et al., 2001).

For optimizing the FNN weights, EAs use two **types of vector representation**: real-valued and binary valued vector representation. In Fig. 4(c), following weight vector representation are illustrated: (1) real-valued coded chromosome, (2) binary coded chromosome, and (3) binary gray-coded chromosome.

Goldberg and Holland (1988) gave the idea of FNN training using GA. However, Whitley (1989) were the first to propose “GENITOR,” a GA based FNN training procedure that used binary-coded chromosome (Fig. 4(c)) for optimizing the weights. Many others followed the idea of GENITOR with some additional improvements such as connectivity optimization and reduced search space introduction (Whitley et al., 1990; Srinivas, and Patnaik, 1991). On the other hand, Belew et al. (1990) used a binary gray coding (Fig. 4(c)) scheme for optimizing the weights, where at first, GA was used for finding initial weights that were further optimized by using BP and vice versa.

The binary bit-string representation of the weights leads to a **precision problem**, i.e., *how many bits would be sufficient for representing weights* and *what would be the total length of a chromosome*. Moreover, the binary representation is computationally expensive because, in each training iteration, a binary to real-valued mapping and vice versa is required. Hence, it is advantageous to use the real-coded chromosome (Fig. 4(c)) directly (Montana and Davis, 1989; Fogel et al., 1990; Sietsma and Dow, 1991; Menczer and Parisi, 1992; Irani and Nasimi, 2011).

Traditional **EA operators** are applied on the binary chromosome. Thus, operators, such as bias-mutation, unbiased-mutation, node-mutation, weight-crossover, and gradient-operator, etc., were defined, for operating on the real-valued chromosome (Montana and Davis, 1989). On the other hand, a matrix-based representation of weights, where a column-wise and a row-wise crossover operators were also defined (Kim et al., 2005). The GA-based real coded weights optimization outperforms BP and its variants for solving real-world applications (Kitano, 1990b; Sexton et al., 1998b; Ding et al., 2011; Tong and Mintram, 2010). Moreover, an evolutionary inspired algorithm, called differential evolution (DE) (Storn and Price, 1997; Das and Suganthan, 2011) that imitates mutation and crossover operator to solve complex continuous optimization problems was found to be performing efficiently for real-valued weight vector optimization (Nolfi et al., 1994; Ilonen et al., 2003; Slowik, 2011).

Similar to DE, **swarm-based or bio-inspired based metaheuristics** directly apply heuristics inspired by nature on a real-valued vector. Hence, they are advantageous in comparison to an EA-based algorithm that needs to simulated mutation and crossover operators for real-valued weight vector (Kennedy et al., 2001). It was found that **PSO** guides a population of the FNN weight vectors towards an optimum population (Ismail and Engelbrecht, 2000; Zhang et al., 2007b). Hence, many researchers resorted to working on swarm based metaheuristics for the FNN optimization.

A *cooperative PSO*, which suggests to splitting a solution vector into n parts, where each part optimized by a swarm of m particles (Van den Bergh and Engelbrecht, 2001, 2004). Thus, an $n \times m$ combinations are constructs an n-dimensional composite vector, where each m swarm contributes to the fitness of a solution. Such cooperation between swarms led to a better performance than that of the basic version of PSO. Similarly, a *multi-phase PSO* was proposed in which particle position was updated only when improvement in location was found; otherwise, the location was copied as-it-is into the next generation (Al-kazemi and Mohan, 2002).

A *cultural cooperative particle swarm optimization* (CCPSO) approach in which a collection of multiple swarms that interact by exchanging information was proposed by Lin et al. (2009). The CCPSO performed better than BP and GA when it was applied for optimizing a fuzzy neural network. Similarly, a hierarchical particle swarm optimization was used to design a beta basis function neural network (Dhahri et al., 2013).

Apart from PSO, there are numerous metaheuristic algorithms among which, some significant metaheuristics were discussed here that were applied for FNN optimization. The continuous version of **ACO** (Socha and Dorigo, 2008) was efficiently applied to optimize the FNN weight vector (Socha and Blum, 2007). ACO trained FNN was found efficient in solving real-life applications, such as scheduling, prediction, image recognition, etc. (Irani and Nasimi, 2012; Sharma et al., 2013).

ABC was efficiently applied on weight vector for optimizing the FNNs (Karaboga et al., 2007; Garro et al., 2011; Sarangi et al., 2014). Similarly, considering the efficiency of **HS** algorithm—that has a slow convergence rate, but guarantees a near-optimum solution (Geem et al., 2001)—many researchers applied HS for optimizing weight vector of the FNNs (Kattan et al., 2010; Kulluk et al., 2012). Moreover, the efficiency of HS comes from using m many harmonies (weight vectors), and iteratively improvising each harmony by computing new harmony (new solution vectors) using heuristic inspired by music pitch modification (Geem et al., 2001; Mahdavi et al., 2007; Pan et al., 2010).

In the past, many **other forms of metaheuristics** were also used for optimizing the FNNs. For example, the application of FF, CS, GSO, BFO, and CFO algorithms for the FNN weights optimization is available in Horng et al. (2012), Vázquez (2011), Ghalambaz et al. (2011), Ulagammai et al. (2007), Zhang et al. (2010), Green et al. (2012) respectively.

Moreover, a comparative study showed that FF algorithm performed better than that of BP, GA, and ABC for weight vector optimization (Nandy et al., 2012). Alba and Marti (2006) provided a detailed study that explains the application of the local and global metaheuristic algorithm for FNN optimization. For example, local search algorithms like SA, TS, GRAP, VNS (Mladenović and Hansen, 1997), estimations of distribution algorithm (Larrañaga and Lozano, 2002) and global search algorithms like GA, ACO, and memetic algorithm, were examined thoroughly by Alba and Marti (2006). Additionally, many researchers studied the performance of metaheuristic algorithms for the training of the FNN and reported that the metaheuristic approaches outperform all the conventional methods by a huge margin (Carvalho et al., 2011; Kordík et al., 2010; Khan and Sahai, 2012).

The **memetic algorithm** supports the hybridization of two or more global metaheuristics for the FNN optimization, which is evident from the following examples. A hybrid GA and PSO approach for optimizing the FNN were proposed by Juang (2004), where both GA and PSO were suggested to run over the same population—randomly generated population W of m individuals (the same individual was treated as a chromosome in GA and a particle in PSO). In each generation of GA and PSO, the fitness of each individual was computed. Then, the best performing individuals (top-half) were marked as elites. The elite individuals were copied to next generation and half of the copied elites were optimized using PSO and the remaining half using GA through tournament selection and crossover operation.

Similarly, a PSO and SA based **hybrid algorithm** for optimizing FNN were proposed by Da and Xiurun (2005), where, in each iteration, each PSO particle was governed by SA metropolis criteria (Metropolis et al., 1953) that determined global best particle for PSO algorithm. There are several other hybrid algorithm examples available in the literature: a hybrid PSO and GA (Ali Ahmadi et al., 2013); hybrid GA and DE (Donate et al., 2013); hybrid PSO and GSO (Mirjalili et al., 2012); and hybrid PSO and optimal foraging theory (Niu et al., 2007).

3.2.2. Architecture plus weight optimization

The basic architecture optimization approach is a *cascade correlation learning*, which iteratively adds nodes to hidden layer to construct optimum architecture (Fahlman and Lebiere, 1990). Moreover, a *constructive* (add node iteratively) and *destructive* (prune nodes iteratively) method (Frean, 1990). However, the constructive and the destructive methods for optimizing architecture are no different from the manual *trial-and-error* method. Therefore, genetic representation of the FNN architecture as mentioned in Figs. 5(a)–(c) can be used for architecture optimization, which is equivalent to searching optimum architecture from a compact space of FNN topology (Maniezzo, 1994; Xi-Zhao et al., 2013).

Let us discuss the genetic representation of architecture in detail. A **direct encoding scheme** (Fig. 5(a)) was proposed by Whitley and Hanson (1989) and Schaffer et al. (1990), where the authors used an adjacency matrix (Fig. 4(b)) to represent connections between nodes, where between any two nodes i and j, a presence of connection is indicated by "1", and absence of connection is indicated by "0". Hence, they were able to encode complete structural information into a chromosome. However, it is disadvantageous because chromosome length increases with network size. Therefore, if only the network's structural information can be encoded into genotype, then, it will avoid chromosome length problem (Harp et al., 1989). Additionally, the encoded network structural information can be accessed using rule-based recursive equation (Mjolsness et al., 1989). Moreover, the represented parametric/structural information into the chromosome can indirectly provide access to the rest of the structural details from a predefined archive (parametric information) (Kitano, 1990a).

The **indirect encoding scheme** reduces chromosome length, where parametric information, such as the number of hidden layers, the number of nodes at hidden layers, the number of connection, etc., makes an archive s. The production rule (Fig. 5(b)) allow us to get access to complete structural information (Fig. 5(c)). Hence, a rule based encoding scheme allows a better FNN architecture optimization than a direct encoding scheme (Siddiqi and Lucas, 1998).

Unlike the weight optimization that has only limited ways of genetic representation, the FNN architecture optimization is an interesting area of research as there are various ways to represent architecture into genotype. It is evident from a fractal configured FNN representation proposed by Merrill and Port (1991), where authors defined each node using parameters, namely, edge code, input coefficient, and output coefficient. Similarly, Andersen and Tsoi (1993) applied GA to evolve each layer separately and Tayefeh Mahmoudi et al. (2013) proposed a grammar encoding and colonial competitive algorithm.

Another approach to the genetic representation of architecture is to encode weights $\mathbf{w}$ (real vector: Fig. 4(c)) and architecture vector $\mathbf{a} = \langle a_1, \ldots a_m \rangle$ (binary vector as Fig. 5(a)) into a combined genotype. Hence, a single solution vector $s = \langle w_1, \ldots, w_n, a_1, \ldots, a_m \rangle$ is obtained (Ludermir et al., 2006), which can be optimized by using metaheuristics.

Many researchers improvised the algorithms itself to optimize architecture. Such examples are as follows: Carvalho and Ludermir (2007) devised a **PSO-PSO** method, in which a PSO (inner PSO block) was applied for optimizing weights that were nested under another PSO block (outer PSO block), that was applied for optimizing the architecture of FNN by adding or deleting hidden node. Similarly, Tsai et al. (2004, 2006) proposed a hybrid Taguchi-genetic algorithm for optimizing the FNN architecture and weights, where authors used a genetic representation of the weights, but they select structure using constructive method (by adding hidden nodes one-by-one). A multi-dimensional PSO approach was proposed by Kiranyaz et al. (2009) for constructing FNN automatically by using an architectural (topological) space. Moreover, the individuals in the swarm population were designed in such a way that it optimized both position (weights) and dimension (architecture) of an individual in each iteration. Thus, optimized FNN weights and architecture simultaneously.

So far, only genetic representation was discussed for evolving architecture. However, **GP** can optimize a phenotype itself, where genetic representation is not required (Khan et al., 2013). Therefore, EP and GP can be directly applied to a population FNN architecture to evolve an optimum FNN architecture (Fogel et al., 1990; Koza and Rice, 1991; Tsoulos et al., 2008).

The design of the FNN architecture is responsible for processing high-dimensional data. Hence, **deep learning** paradigm offers study massive and deep structure of the neural network that can process complex problems related to speech processing, natural language processing, signal processing, etc., (Schmidhuber, 2015; LeCun et al., 2015). Such a variant of the FNN is *convolutional neural networks* (ConvNets), which is designed to process data from the multiple arrays form such as a color image composed of three 2D arrays (Schmidhuber, 2015; LeCun et al., 2015). The ConvNets has a three-dimensional

(i) $\langle 0110\ 110\ \ 01\ 1 \rangle$

(ii) $\langle \mathbf{0}0110\ 0\mathbf{0}110\ 000\mathbf{0}1\ 000\mathbf{0}1\ 0000\mathbf{0} \rangle$

(a)

$$S \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

$$A \rightarrow \begin{bmatrix} z & u \\ z & z \end{bmatrix}, B \rightarrow \begin{bmatrix} z & z \\ c & z \end{bmatrix}, C \rightarrow \begin{bmatrix} z & z \\ z & z \end{bmatrix}, D \rightarrow \begin{bmatrix} z & z \\ z & z \end{bmatrix}.$$

$$u \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \ z \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, c \rightarrow \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix},$$

$$a \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \ i \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

(b)

$$S \rightarrow \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(c)

Fig. 5. Mapping of phenotype (Fig. 4 to genotype (for architecture). (a) Direct encoding to a vector of connectivity matrix (Fig. 4(b)): (i) upper right triangle; (ii) complete connectivity. (b) Indirect encoding schemes for architecture (Fig. 4(a)), where S is a start symbol, A, B, C, and D are the variables, and a, c, i, and u is the terminal. (c) Complete connectivity derived from rules operation shown in Fig. 5.

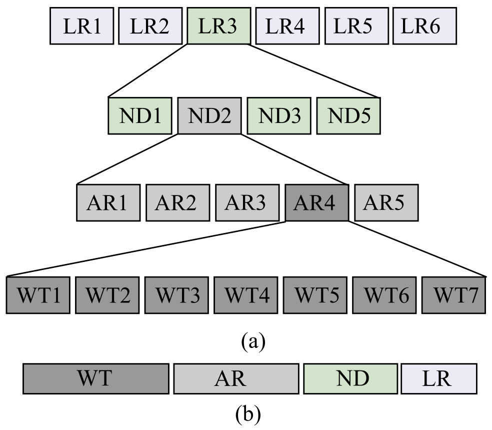


Fig. 6. Meta-learning scheme (a), where LR is learning parameter, ND is activation function, AR is architecture, and WT is weight (Abraham, 2004). Combined chromosome structure (b).

arrangement of neural nodes. Hence, it efficiently receives 3D inputs and processes them to produce 3D outputs (Maturana and Scherer, 2015).

In contrast to deep network paradigm, an **extreme learning machine** (ELM) based hierarchical learning framework (H-ELM) proposed by Tang et al. (2016) claimed a faster learning than deep learning by ELM (Huang, 2014) based auto-encoding. The proposed H-ELM framework worked in two phases: unsupervised hierarchical feature representation and (2) supervised feature classification (Tang et al., 2016).

3.2.3. *Input layer optimization*

Input layer optimization resembles feature reduction, which is traditionally performed separately by dimensionality reduction methods (Fodor, 2002). However, reducing input dimension by optimizing input layer, i.e., by feeding a subset of input features at the input layer than by feeding the whole set of input features enhances FNN's performance (Fontanari and Meir, 1991; Guo and Uhrig, 1992). Therefore, FNN has a functional dependency on the problem at hand.

EAs select a subset of input features for which FNN perform better than that of the complete feature set (Fontanari and Meir, 1991). For this purpose, a genetic representation of input features is required in which the available features are placed on a genetic strip, and the presence of a feature is marked as "1" and the absence of a feature is marked as "0." Such mechanism of input layer optimization was found advantageous in improving NN performance (Venkadesh et al., 2013).

Moreover, binary PSO (Kennedy and Eberhart, 1997), which is a **discrete optimization method** was employed for selecting the input features which were binary coded (Lin et al., 2008; Vieira et al., 2013). Lin et al. (2008) proposed a modified version of binary PSO, where EA like mutation operator was applied to mutated binary vectors. Similarly, ACO, which traditional solves discrete optimization problem was applied to select input features and training of an FNN in a hybrid manner (Sivagaminathan and Ramakrishnan, 2007).

Input layer optimization which is related to input feature reduction can also be thought as **training data optimization**. Training data optimization is helpful, particularly when data is insufficient or noisy. Zhang and Veenker (1991) resorted to performing an adaptive selection of input examples by employing genetic selection, where two-point and one-point crossover operations created new example patterns. For the crossover operations, the parent's examples were drawn from the original input set. Also, mutation operators were also applied for generating new child example. Hence, the efficiency of FNN was improved when trained over the modified new examples.

Additionally, Cho and Cha (1996) suggested an **input example generation** methods, in which the input space was divided into many regions, and *k*-nearest neighbor method was applied to determine/generate a new virtual example, mainly for the sparse region of the input space. Hence, both the above methods of input example generation sought to enrich knowledge space for the FNN learning (Zhang and Veenker, 1991; Cho and Cha, 1996).

3.2.4. *Node optimization*

Primarily, node optimization can be addressed in three ways: (1) by choosing activation functions at the FNN active nodes from a set of activation functions (Liu and Yao, 1996; Tong and Mintram, 2010); (2) by optimizing the arguments of activation function (Ojha et al., 2014); and (3) by placing a complete model at the nodes of a network (Oh and Pedrycz, 2002; Hirose, 2006).

It was found that FNN performed better when it has **non-homogeneous nodes** (different activation function at different nodes) than that of the homogeneous nodes (Mani, 1990). Liu and Yao (1996) aimed for an evolution in FNN nodes by selecting sigmoid and Gaussian function adaptively at the nodes. Moreover, adaptation in both nodes and architecture using EAs, where the design of nodes was inspired by locus flight system and tailflip of crayfish (Dumont et al., 1986), can further improve FNN performance (Stork et al., 1990). For this purpose, Ling et al. (2007) gave idea of an input dependent FNN that had a combined chromosome representation (Fig. 6), where a real-coded GA for simultaneous optimization of weights, activation functions, and architecture was used.

On the other hand, to optimize nodes, a **family competitive EA** was proposed by Yang and Kao (2001), where three operators, such as decrease-Gaussian-mutation, Cauchy-mutation, and self-adaptive-mutation were defined. Moreover, family-competition is a process that generates a pool of *L* many FNNs by recombination and mutation operations and selects an FNN from that pool. The family-competition with different mutation operator is repeated until the best FNN is found. Many others found that the adaptation in FNN nodes by one of the methods mentioned above can improve FNN performance to some extent (Alvarez, 2002; Leung et al., 2003; Augusteijn and Harrington, 2004; Nedjah et al., 2007).

The third aspect of node optimization is to design a node as a model itself. Such modification leads to variate of neural network paradigms such as *polynomial neural network* (Oh and Pedrycz, 2002; Andoni et al., 2014), where the nodes are designed to as a polynomial function based on inputs to the nodes. Similarly, the nodes of a *GMDH neural network* is designed as an Ivakhnenko polynomial (Puig et al., 2007); the nodes of a *complex value neural network* or *multivalued neural network* is designed with a complex value activation functions (Hirose, 2006); the node of *spiking neural networks* has specific behavior, in which a node signal is propagated to another node only if the intrinsic quality of neural activation value is above a defined threshold (Sporea and Grüning, 2013); the nodes of *fuzzy neural network* paradigm is designed using the concepts of fuzzy theory (Fullér, 2013); the node and the architecture of the *Quantum neural network* are inspired by the quantum computing (da Silva et al., 2016; Narayanan and Menneer, 2000; Kouda et al., 2005; Li et al., 2013). In all such methods, metaheuristics have a significant role in the optimization.

3.2.5. *Learning algorithm optimization*

The initial thought of learning algorithm optimization is the optimization of its parameters. For example, the optimization the learning rate and the mutation factor parameters of BP by applying some metaheuristics (Belew et al., 1990). To optimize the parameters of an FNN learning algorithm, its parameters (e.g., BP parameters) and learning rules are encoded onto a genotype (Harp et al., 1989; Baxter, 1992). However, formulating BP parameter such as learning rule, which is a dynamic concept, into a static chromosome is disadvantageous (Chalmers, 1990). Hence, a genetic coding for four components (current weight, activation function of the incoming node and outgoing nodes, input) local to weight in an FNN can encode (Chalmers, 1990).

Moreover, assuming that each node in a network uses same learning rule, an evolution in learning was proposed in Kim et al. (1996), where weights optimization related to a particular node depended only on the input/output at that node. Evolution in learning rule can be described as described by Yao (1999):

$$\Delta \mathbf{w}^t = \sum_{k=1}^{n} \sum_{i_1, i_2, \ldots, i_k = 1}^{n} \left(\theta_{i_1, i_2, \ldots, i_k} \prod_{j=1}^{k} w_{ij}^{t-1} \right) \tag{10}$$

where t is the iteration, $\Delta \mathbf{w}^t$ is weight change, $w_1, w_2, \ldots, w_n$ are weights associated with a node, θ_i is real-valued coefficient that is determined by using EAs. However, learning rule (10) is impractical because of it required huge computation time. Hence, bio-inspired algorithms may be employed for determining the coefficient in (10).

3.2.6. *Combination of FNN components optimization*

Fig. 3 is an impressive representation of the most of FNN optimization combinations, where the genetic representation of the combination of FNN components can be represented in Fig. 6, which refers to a hierarchy of combination, called meta-learning scheme. In the meta-learning scheme, top down or bottom up optimization approach, which means, weights to learning rule and learning rules to weight optimization can be adopted (Abraham, 2004). However, it resembles one-by-one learning scheme. Hence, the advantageous approach is to represent each component of FNN side-by-side onto a genetic vector for optimization, which indicates the confluence of all components of FNN as indicated by area "a8" in Fig. 3.

Yao (1993, 1999) summarized all such form of adaptation in **evolutionary artificial neural network** (EANN), which is a special class of artificial neural network, where in addition to learning; evolution is another fundamental form of adaptation. Infact a paradigm, called *Neuroevolution* that accommodates adaptive learning all or some components of FNN in some intuitive ways by applying EAs. For examples, generalized acquisition of recurrent links (GNARL) (Angeline et al., 1994), evolutionary programming net (EPNet) (Yao and Liu, 1997), neuroevolution of augmenting topologies (NEAT) (Stanley and Miikkulainen, 2002), hypercube-based neuroevolution of augmenting topologies (HyperNEAT) (Gauci and Stanley, 2007), evolutionary acquisition of neural topologies (EANT2) (Kassahun and Sommer, 2005), and heterogeneous flexible neural tree (HFNT) (Ojha et al., 2017) optimizes both FNN structure and parameters (weights) using some direct or indirect encoding methods. Moreover, several other paradigms and methods proposed in the past for the simultaneous optimization of FNN components are described as follows.

A **structured genetic algorithm** was proposed by Dasgupta and McGregor (1992), which simultaneously optimized both architectures and weights. It was found that the simultaneous optimization of both weight and architecture lead to a better generalization (Kitano, 1994; Maniezzo, 1994; Girosi et al., 1995; Arifovic and Gencay, 2001). Considering permutation[2] problem in a GA, EP-based mutation mechanism for evolving FNN architecture was proposed by Yao and Liu (1997) is known as EPNet.

A **neuroevolution of augmenting topologies** (NEAT) introduced by Stanley and Miikkulainen (2002) was a GA-based evolution of an FNN phenotype as a whole, in which a special mutation and crossover operator were defined for manipulating nodes and connections of FNN. Specifically, the linear network information FNN weights, nodes, and connection information were encoded using genetic encoding. The proposed NEAT was evaluated over several applications, and its performance was found outperforming static FNN topology.

A **virtual subpopulation** approach was proposed by Salajegheh and Gholizadeh (2005) for the optimization of FNN using EAs. Later, while indicating a permutation problem, crossover operator as a combinatorial optimization problem was proposed in which each hidden node was considered as a subnetwork and a complete network was evolved using the evolution of several subnetworks (García-Pedrajas et al., 2006). Additionally, GA-based and SA-based crossover operators were applied to generate an offspring (new individual subnetwork). To maintain diversity in population, two mutation operators such as BP-mutation and random-mutation were proposed. In BP-mutation, few iterations of BP algorithm were applied to update weights of the subnetwork, and in random mutation, weights of subnetwork were randomly replaced with new weights. Hence, a *coevolution* of FNN weights and architecture was proposed that evolved FNN with the *cooperation* of the individuals of a subnetwork population.

A **cooperative coevolution neural network** process—inspired by virtual subpopulation approach (Salajegheh and Gholizadeh, 2005)—was proposed by Moriarty and Miikkulainen (1997), which was a symbiotic, adaptive neuroevolution (SANE) algorithm for constructing FNN in a dynamic environment. Unlike conventional evolutionary approach, which uses a population of FNNs, SANE uses a population of nodes, where each node establishes connections with the other nodes to form a complete network.

Two reasons of better performance of SANE over conventional and stand-alone metaheuristics were suggested. First, since SANE consider the nodes as functional components of the FNN, it accurately searches and evaluates nodes as genetic building blocks. Second, since a node alone cannot perform well and evolutionary process evolves different types of nodes, SANE was able maintains diversity in the population. Later, the concept of SANE was extended, in which the selection of several individuals from a population of hidden nodes was combined in a various permutation in order to form several complete networks, i.e., evolution in hidden nodes led to an evolution of the complete network (Garcia-Pedrajas et al., 2003).

A concept of **sparse neural trees**, in which GP for evolving network structure and GA for parameter optimization was suggested by Zhang et al. (1997). Similarly, a *flexible neural tree* (FNT) concept, where GP was used for the adaptation in network structure and SA for the optimization of the parameters (including parameters of activation function) was proposed by Chen et al., (2004, 2006). FNT is a tree-like model where adaptation in all components of is equally important. Moreover, its components adaptation may take many forms (Fig. 3). Hence, a beta basis function—which has several controlling parameters, such as shape, size, and center—was used at non-leaf nodes of an FNT (Bouaziz et al., 2014). It was observed that embedding beta-basis function at FNT nodes has advantages over other two parametric activation function. A parallel evolution of FNT using MPI programming and GPU programming respectively were proposed by Peng et al. (2011) and Wang et al. (2012).

A slightly different direction of FNN modification and improvement study can be seen as the study of **quantum neural network** (QNN). At the first place, the QNN as a *quantum perceptron* was proposed by Lewenstein (1994), where instead of classical weights, a unitary operator was used to map inputs to an output. The study in QNN encompasses the development of quantum weights, quantum neurons, quantum network, and quantum learning (Narayanan and Menneer, 2000).

The design/algorithm of quantum network was thought as an algorithm that can find the control parameters for a coupled qubit system (Gershenfeld and Chuang, 1998) as it appears in quantum computing. A comprehensive quantum inspired neural network is presented by Menneer and Narayanan (1995), where two categories of inspiration were drawn: strongly and weakly quantum inspired FNN. In strongly inspired QNN, each pattern in a training set was considered as a particle which is processed by a number of FNNs in different universes. Such process was compared with the electrons or photons passing through many slits simultaneously. Whereas, in weakly inspired QNN,

[2] Permutation problem occurs when using traditional crossover operator, where a population has traditional genetic representation of FNN architecture.

each training pattern (though as a particle) was in its own universe. Moreover, there were various QNN models proposed in the past by (1) Behrman et al. (1996), (2) Chrisley (1997), (3) Menneer and Narayanan (1995), and Ventura and Martinez (1998). A detailed description of these QNN models is offered by Rudolph (2011).

3.3. Comments on metaheuristics approaches

It is indeed can be concluded that metaheuristic algorithms have provided various dimension to the optimization of the FNNs. It has opened several ways such that a generalized FNN can be obtained. Especially the architecture simplification which is directly related to the generalization of an FNN can easily be achieved through the evolving FNN together with its other components. However, the primary disadvantage of using metaheuristic algorithms is the training time consumption. Since the metaheuristic algorithms use a population (many solution candidates) during optimization, the time consumption becomes directly proportional to the number of candidates in a population.

It can also be argued that both conventional and metaheuristic based FNN training take by far more training time than the *extreme learning machine* (ELM) (Huang et al., 2006b). ELM is a three-layered FNN architecture whose weights between input and hidden layer are randomly assigned and never updated. Additionally, the weights between hidden and output layer are updated in a single step using some least square estimation. Hence, extremely less time required for the learning of the FNN.

It is stated in NFL theorem (Wolpert, 1996) that it is difficult to find a metaheuristic algorithm that solves all class of problems. Hence, a metaheuristic algorithm may find difficulty in optimizing the FNN that has been formulated for solving some specific problem (input patterns). Additionally, it is not theoretically possible to understand or determine that how fast a metaheuristics algorithm will converge or finds a satisfactory solution. The only way to determine a metaheuristic algorithm's convergence is by its empirical evaluations. Moreover, since each metaheuristic applies some specific heuristic, it is difficult to select one metaheuristic as the best metaheuristic at an instance for a problem. It is only possible to select a metaheuristic by empirically comparing the convergence speeds and trained FNN performances.

4. Multiobjective metaheuristic approaches

Multiobjective optimization procedure involves in optimizing two or more objectives functions, simultaneously. Multiobjective algorithms are efficient methods for evaluating Pareto-optimal solutions for multiobjective problems. Since optimizing training error cannot provide generalization alone, FNN optimization is viewed from the multiobjective perspective.

Let us first investigate: *why the multiobjective framework is needed for FNN optimization, what are the objective functions required for framing FNN as a multiobjective problem,* and *how the objective functions can be framed into multiobjective optimization.* Answers to these question lie in the following discussion.

First, a cost function (4) or any equivalent function is the foremost necessity for the supervising training of FNN.

Second, the generalization of FNN is an essential aspect of its optimization. One approach is to use validation error on cross-validation data because an FNN with low training error may not perform well on unseen (test) data unless FNN is generalized. Moreover, minimization of generalization error is essential than the minimization of training error.

Another approach is to add a regularization term to the training error to avoid overfitting. Additionally, minimizing network complexity leads to a better generalization (Jin et al., 2005a). Hence, generalization can be achieved by adding a complexity indicator term to training error, i.e., the generalization by minimizing training error and simplifying network complexity.

Third, reducing input-feature—when a problem is available with a huge input dimension (feature)—can lead to a better generalization. However, input dimension reduction and training error reduction are two contradictory objectives.

Finally, the conclusion is, the training error (4) or equivalent cost function c_f needs to be optimized with one or more additional objectives to achieve generalization, which is why multiobjective framework for optimizing FNN are used.

Let us say that training error (4) is c_{trn}, and an additional objective is c_{add}. So, a generalized error c_{gen} may be computed by adding an objective to training error as:

$$c_{gen} = c_{trn} + \lambda c_{add}, \tag{11}$$

where $\lambda > 0$ is a hyperparameter that controls the strength of additional objective c_{add}. The validation error term c_{cv}, regularization term c_{reg}, or network complexity c_{net} or a combination of all can be considered as an additional objective c_{add} in (11). The regularization term c_{reg} is the weight decay or norm of weight vector $\mathbf{w}$ as:

$$c_{reg} = \frac{1}{2}\sum_{i}^{n} w_i^2 = \frac{1}{2}\left\|\mathbf{w}\right\|^2. \tag{12}$$

Similarly, a validation error c_{cv} is usually computed using (4) on a cross-validation data. On the other hand, the network complexity c_{net} is computed as:

$$c_{net} = \sum_{i}^{z}\sum_{j}^{z} c_{ij}, \tag{13}$$

where z is the number of nodes in the network c (Fig. 5(a)), or any user-defined function can also be used for evaluating network complexity, e.g., the number of nodes, the number of connections, etc.

However, the generalization objective of the form (11) is a scalarized objective that has two disadvantages (Das and Dennis, 1997). First, determining an appropriate hyperparameter λ that controls the contradicting objectives. Hence, the generalization ability of the produced model by using (11) will be a mystery. Second, the objective (11) leads to a single best model that tells nothing about how contradicting objectives were handled. In other words, no single solution exists that may satisfy both objectives. Therefore, generalization error (11) need to be formulated into a multiobjective form: $\min\{c_{trn}, c_{reg}, c_{cv}, \ldots\}$, i.e., a multiobjective optimization needs to be performed as:

$$\text{minimize } \{c_1(\mathbf{w}), c_2(\mathbf{w}), \ldots, c_m(\mathbf{w})\}\text{subject to } \mathbf{w} \in S,$$

where $m \geq 2$ is the number of objective functions $c_i: \mathbb{R}^n \to \mathbb{R}_{\geq 0}$. The vector of objective functions is denoted by $\mathbf{c} = \langle c_1(\mathbf{w}), c_2(\mathbf{w}), \ldots, c_m(\mathbf{w})\rangle$. The decision (variable) vectors $\mathbf{w} = \langle w_1, w_2, \ldots, w_n\rangle$ belong to the set $S \subset \mathbb{R}^n$, which is a subset of the decision variable space $\mathbb{R}^n$. The word "minimize" indicates the minimization all the objective functions simultaneously.

A nondominated solution is one in which no one objective function can be improved without a simultaneous detriment to at least one of the other objectives of the solution. The nondominated solution is also known as a Pareto-optimal solution.

Definition 3. Pareto-dominance - A solution $\mathbf{w}_1$ is said to dominate a solution $\mathbf{w}_2$ if $\forall\, i = 1, 2, \ldots, m,\ c_i(\mathbf{w}_1) \leq c_i(\mathbf{w}_2)$, and there exists $j \in \{1, 2, \ldots, m\}$ such that $c_j(\mathbf{w}_1) < c_j(\mathbf{w}_2)$.

Definition 4. Pareto-optimal - A solution $\mathbf{w}_1$ is called *Pareto-optimal* if there does not exist any other solution that dominates it. A set *Pareto-optimal* solution is referred to as a Pareto-front.

Now, a multiobjective algorithm must provide a homogeneous distribution of a population along Pareto front and improve solutions along successive generations (Gaspar-Cunha and Vieira, 2005). Hence, three basic operators can be used (Gaspar-Cunha and Vieira, 2005). (1) *Fitness assignment* to guide a population in the direction of Pareto-front using robust and efficient multiobjective selection method. (2) *Density estimation* to maintain solutions distributed over entire Pareto-front using operators that take account of the solution's proximity. (3) *Archiving* to prevent degradation in fitness during successive generations by maintaining an external population to preserve the best solutions and for periodic input to the main population. A detailed survey of multiobjective algorithms is offered by Coello (1999) and Zhou et al. (2011). Now, based on the above discussion on the cost functions and generalization conditions, the multiobjective for FNN optimization can be categorized as *non-Pareto based multiobjective* approach and *Pareto-based multiobjective* approach.

4.1. Non-Pareto based multiobjective approaches

In non-Pareto based multiobjective approach, the objective functions are aggregated as mentioned in (11) or by some other means. For example, de Albuquerque Teixeira et al. (2000) proposed to add a regularization term $c_0 - c_{reg}$ to training error $c_0 - c_{trn}$, where c_0 is the origin of the two objectives. To obtain an efficient solution, they designed a vector **vc** of **scalar objectives** by varying the hyperparameter λ from 0 to 1. Hence, training FNN for each scalar objective in vector **vc**, a Pareto set was obtained, and then, it was possible to select the best solution from the Pareto front. However, this was an expensive approach, which does not use any Pareto-based multiobjective algorithm to compute Pareto set; rather, an ellipsoid method (Bland et al., 1981) was applied to train FNN for each scalar objective $vc_i \in \mathbf{vc}$ sequentially.

Similarly, to achieve generalization, Costa et al. (2003) proposed a **sliding mode control BP** algorithm for the multiobjective treatment to FNN objectives c_{trn} and c_{reg}. The optimization trajectory of the 2D space of the objectives c_{trn} and c_{reg} was controlled by modifying BP weight update rules using two sliding surface control indicators each belongs to the mentioned objectives, respectively.

Multiobjective treatment to FNN was also offered by using improvising metaheuristics itself such as a *predator-prey* algorithm was devised by Pettersson et al. (2007). To get a generalized network, the predator-prey algorithm used a family of the randomly generated population of sparse neural networks, called *pray population* and an externally induced family of *predators population* whose job was to prune preys populations based on the objectives c_{trn} and c_{net} was also generated. Similarly, a hybrid multiobjective approach, where a geometrical measure based on singular-value-decomposition for estimating a necessary number of nodes in a network was proposed by Goh et al. (2008).

Additionally, a micro-hybrid genetic algorithm was introduced to fine-tuning the network performance. A hybrid algorithm, which uses GA for evolving FNN and uses PSO, BP, and LM for fine-tuning the evolved FNN was proposed by Almeida and Ludermir (2010). In the proposed hybrid algorithm, several objectives function such as training error c_{trn}, validation error c_{cv}, number of hidden layers c_{hid}, number of nodes c_{node}, and activation function c_{fun} were aggregated as:

$$c_{net} = \alpha c_{trn} + \beta c_{cv} + \gamma c_{hid} + \delta c_{node} + \theta c_{fun}, \tag{14}$$

where, α, β, γ, δ, and θ were controlling parameters. Hence, multiple objectives were optimized simultaneously.

As mentioned above in Section 4, the **aggregating objective** function has disadvantages in obtaining the best generalized solution. It is evident from (14) that determining hyperparameters for controlling objective function is a challenging task. Therefore, Pareto-based multiobjective is an efficient choice for the multiobjective treatment of FNNs.

4.2. Pareto based multiobjective approaches

The advantages of applying Pareto-based learning is thoroughly explained and compared with a single and scalerized objective by Jin and Sendhoff (2008). For example, a **nondominated sorting genetic algorithm version II** (NSGA-II) (Deb et al., 2000) when used for optimizing objectives c_{trn} and c_{net} offers a Pareto set by optimizing both objectives simultaneously using a nondominated sorting method as defined in Definition 3. Hence, NSGA-II can be applied to obtained a regularized network by optimizing the objectives c_{trn} and c_{reg} (Jin et al., 2004).

Similarly, **Pareto differential evolution** (PDE) algorithm and its variant self-adaptive PDE algorithm was applied to optimize objectives c_{trn} and c_{net} simultaneously that offered a Pareto-set, from which the best solution was picked-up according to network complexity and approximation error examination (Abbass, 2003, 2002). Simultaneous optimization of the objectives c_{trn} and c_{net} were also addressed using **multiobjective PSO** to generalize FNN performance (Yusiong and Naval, 2006).

For an image classification problem, Wiegand et al. (2004) pointed out two crucial points: the classification speed and the classification accuracy c_{acc}. The classification speed was then related to the network complexity (number of hidden neurons) c_{net}. The proposed trade-offs between classification speed and classification accuracy were addressed using NSGA-II.

Similarly, Roth et al. (2006) studied three methods for image classification problem: linear aggregating (LA), NSGA-II with deterministic selection (DM), and NSGA-II with tournament selection (LM). They proposed to optimize network complexity c_{net} and accuracy c_{acc}. Moreover, they combined regularization term c_{reg} with accuracy c_{acc} and proposed an adaptive strategic for designing network topology using reproduction operators for both hidden layer and input layer. The hidden layer operators were add-connection, delete-connection, add-node, and delete-node. The receptive (input) layer had the following operators: add-connection, delete-connection, add-node, and delete-node. Interestingly, they observed that DM and LM performed better than LA, i.e., Pareto-based multiobjective algorithms performed better than that of the scalerized objectives. Such ability of the Pareto-based treatment to FNN to obtain general FNN was exploited by several researchers for solving many real-life applications (Jin et al., 2005b; Furtuna et al., 2011; Zăvoianu et al., 2013; Karpat and Özel, 2007).

Further, the **coevolution FNN** concept (Garcia-Pedrajas et al., (2003, 2002b)) was extended by García-Pedrajas et al. (2002a) under the multiobjective framework, by using *subnetwork* and *network* concepts. A subnetwork was a collection of nodes, i.e., a subnetwork was considered as a hidden node for a network. Therefore, a network was a collection of subnetworks. So, a population P_1 of subnetwork, which was evolved separately using NSGA-II was used to construct a population P_2 of networks. Then, NSGA-II was again applied to evolve population P_2. Interestingly, authors defined separate objectives for population P_1 (subnetworks objectives) and P_2 (networks objectives) so that the functional diversity in both network and subnetwork can be maintained. Additionally, some metrics (objectives) for measuring network and subnetwork functional diversities were defined. The objective of subnetworks were *differences* (for maintaining functional diversity of subnetwork), *substitution* (to replace poor candidates by better candidates), and *complexity* (for counting the number of connection, nodes, and sum of all weights). Therefore, they coevolved overall network with the cooperation of subnetwork that evolves together with the whole network to get a general solution to a problem.

Apart from the discussed objective in this Section, some interesting dimensions in multiobjective treatment to FNN is offered by Giustolisi and Simeone (2006), in which they proposed to apply NSGA-II for the simultaneous optimization of three objectives: *input-dimension*, training error, and network complexity. Hence, an optimized a network that performs well on the minimal set of input dimension was obtained. Similarly, Cruz-Ramírez et al. (2010) and FernandezCaballero et al. (2010) used a Pareto-based memetic algorithm approach for combining PDE and Rprop algorithms to minimize objective pairs *true* *classification rate* and *minimum sensitivity* (miss-classification rate), simultaneously.

As a result of metaheuristic or multiobjective metaheuristic treatment, a set of FNN network is obtained and selecting the best FNN from that set is a difficult task. Since selecting a single best FNN from may not offer a generalized solution and the residual error can still be remaining many problems when selection single best FNN (Sejnowski and Rosenberg, 1987), then an ensemble of a set of FNNs is recommended.

5. Ensemble of feedforward neural networks

Metaheuristics optimization of FNN leads to a final population that contains many solutions close to the best solution. Moreover, the solution in the final population are divers in the following sense: (1) parametric (each FNNs have different sets of weights); structural (each FNNs have different network configurations); and (3) training set (each FNNs are trained on different parts of a training set). Hence, a collective decision (ensemble) of/many diverse candidates selected from a final population may offer desired generalization (Yao and Liu, 1998b). The literature that explains *how to construct diverse FNNs* and *how to combine decisions of diverse FNNs* are summarized as follows.

The very basic idea is to apply single-solution based algorithms on/many FNNs to get/many diverse solutions (Hansen and Salamon, 1990). The decision of/many candidates which were created either by single solution-based metaheuristics, or by population-based metaheuristics, or by any other means are combined using the following methods (Polikar, 2006; Yao and Liu, 1996): (1) majority voting method (for classification problems); (2) arithmetic mean (for regression problem); (3) rank-based linear combination; (4) linear combination by using *recursive least square* (Davis and Vinter, 1985) (to minimize weighted least squares error so that redundant individuals are eliminated); (5) evolutionary weighted mean or majority voting (metaheuristic to determine impact of an FNN in ensemble); and (6) entropy-based method for combining FNNs in ensemble (assigning entropy to FNNs during the learning process) (Zhao and Zhang, 2011).

Since population-based metaheuristics lead to an optimized final population, it is advantageous to use the final population for making ensemble (Yao and Liu, 1998b). However, there are two fundamental problems with it (Liu and Yao, 1999): (1) determining ensemble size, (2) how to maintain diversity in the population. Hence, a **negative correlation learning** (NCL) algorithm that optimized and combined individual FNNs in an ensemble during learning process was proposed by Liu and Yao (1999). NCL optimized all individual FNNs simultaneously and interactively by adding a correlation penalty terms to the cost functions. Moreover, NCL produced negatively correlated and specialized FNNs by using co-operation among each FNNs of a population (Qin et al., 2005).

To determine the size of ensemble automatically, EA-based ensemble procedure was laid down in which NCL was applied during networks training. Moreover, different FNNs were allowed learn different parts of training data and the best (according to fitness) were selected for ensemble (Liu et al., 2000). Additionally, a **constructive-cooperative-neural-network-ensemble** was proposed by Islam et al. (2003) that determined ensemble size by focusing on accuracy and diversity during a constructive, cooperative procedure (Yao and Islam, 2008).

However, mere training fitness based selection of candidates for the ensemble is insufficient because it does not tell much about candidates role/influence in the ensemble. This problem was addressed in a **GA-based selective ensemble** method (Zhou et al., 2002), which selects a subset of the population and determine the strength of selected candidates using GA. It was also shown that the ensemble of a subset of the population was found performing better than that of the whole population (Zhou et al., 2002). The effectiveness such GA-based selection was found efficient than the traditional ensemble methods: *bagging* (Breiman, 1996) and *boosting* (Schapire, 1990).

It is beneficial to partition/fracture training data and allows different FNN in the population to learn various parts of training data (Breiman, 1996; Schapire, 1990). An evidence of such was examined by Bakker and Heskes (2003) and Chen et al. (2010), where it was found that the ensemble of a few FNNs that was trained using *bootstrapping* performs better than that of an ensemble of a larger number of FNNs. Similarly, the efficiency of using distinct training sets for optimizing different FNNs was proved when a **class-switching ensembles** approach proposed by Martínez-Muñoz et al. (2008) and were compared with bagging and boosting methods.

At one hand bootstrapping method allows FNN to learn different training samples. On the contrary, a **clustering-and-coevolution** approach for constructing neural network ensembles proposed by Minku and Ludermir (2008) partition the input space using a clustering method to reduced number of input nodes of FNNs. Hence, in the ensemble, diverse FNNs (different FNNs were specialized in various regions of input space) were created. Moreover, it reduced run time of learning the process by coevolving (divide-and-conquer method) different FNNs using cooperation between FNNs. Such method improves diversity and accuracy of an ensemble system (Minku and Ludermir, 2008).

Similarly, a method was suggested by Kim and Cho (2008) for generating diverse evolutionary FNNs using a fitness-sharing method—a fitness sharing method shares resources if the distance between the individuals is smaller than the predefined sharing radius. Specifically, authors proposed a speciation based evolutionary neural ensemble method for constructing ensemble by combining FNNs using a knowledge space method. On the other hand, a progressive interactive training scheme called a **sequential-neural-network-ensemble-learning** method, which trained FNNs one-by-one by interaction from a central buffer of FNNs was proposed by Akhand et al. (2009).

Both diversity and accuracy is a crucial aspect in construing ensemble of FNNs (Polikar, 2006). However, accuracy and diversity are contradictory to each other, so, a multiobjective approach may be applied to evolve FNN population by maintaining accuracy and diversity simultaneously (Chandra and Yao, 2006). For this purpose, **multiobjective regularized negative correlation learning** that maximized performance and maximized the negative correlation between individuals in population was found efficient (Chen and Yao, 2010).

6. Challenges and future scopes

The effectiveness of FNN training primarily depends on *data quality*, which is governed by the following *data quality assurance* parameters: accuracy, reliability, timeliness, relevance, completeness, currency, consistency, flexibility, and precision (Wand and Wang, 1996; Pipino et al., 2002). Usually, *data cleaning* is a major step before modeling (Hernández and Stolfo, 1998). Therefore, training of the FNN remains always sensitive to the data cleaning process and it poses a significant challenge to adapt some mechanism in training process such the sensitivity towards data-clean may be reduced. Additionally, one problem related to data-driven modeling (FNN learning) is the data itself which can be insufficient, imbalanced, incomplete, high-dimensional, or abundant.

For the case **insufficient data**, usually the input hyperspace is exploited to generate virtual samples to fill the sparse area of the hyperspace, and by monitoring FNN performance on the virtually generated samples (Cho et al., 1997). The second approach exploits the dynamics of EAs in conjunction with FNNs to obtain new samples (Zhang and Veenker, 1991). However, this area is still much to explore, where some open questions such as how efficiently FNNs can be trained with virtually generate data to mitigate the insufficiency. On the other hand, research in the area of imbalance dataset is continued to interest researcher (Mazurowski et al., 2008).

The present era of data analysis is what we call *big data*, i.e., we need to deal not only with *high-dimensional data* but also with the *variety data* and **stream data** (Zikopoulos et al., 2011). High-dimensional data, such as gene expression data, speech processing, natural language processing, social-network-data, etc., poses significant challenges. Such challenge is to some extent addressed by *deep learning* paradigms that allow the arrangement several units/layers of FNNs (or any other model) in a hierarchical manner to process and understand insights of such high-dimensional data (Hinton et al., 2012, 2006). High-dimensional data can also be managed/reduced by encoding or decoding methods and using FNNs training (Hinton and Salakhutdinov, 2006). Therefore, FNNs has a greater role in feature reduction.

In a **non-stationary environment**, such as stock-price market, weather forecasting, etc., data comes in the stream, i.e., data comes in sequential order, and traditionally, re-training based mechanics for dynamic learning (online learning) of FNN is the basic option (Saad, 2009). However, it is still an open problem to design strategies for the dynamic training of FNN.

Apart from the crucial aspects that *how to manage non-stationary data*, the aspect that how to handle **multi-view (heterogeneity)** of data is an additional challenge. The quest of developing a model that can stand robust and efficient for the non-stationary data caused by the time-dependent process of data generation, and can accommodate new knowledge (newly generated data sample) is a significant topic in machine learning research (Ditzler et al., 2015). On the other hand, integration of data or of the models for that matter for the heterogeneous data generated or gathered from different instruments and data-generation processes is a significant research problem (Ritchie et al., 2015; Pavlidis et al., 2001).

Moreover, present era, the **fourth industrial revolution**, is of *Internet of Things* (IoT) (Prisecaru, 2016). In IoT, sophisticated technologies such as *smartphone* and *smartwear* provide several forms of data, e.g., *human activity recognition* (Kim et al., 2010). Additionally, it demands application to be simple. Hence, FNN models which when aims to such technologies needs to be less complex. Therefore, FNN architecture simplification or model's complexity reduction is a challenging task. Such problem can be addressed through the integration of FNN with statistical methods like the one usually done with *hidden Markov model* (Trentin and Gori, 2001). Therefore, such kind of modification to network architecture and specialized node design may lead to different paradigms of FNN that may solve various real-world complex problems.

7. Conclusions

Feedforward neural network (FNN) is used for solving a wide range of real-world problems, which is why researcher investigated many techniques/methods for optimizing and generalizing FNN. Specifically, metaheuristics allow us to innovate and improvise methods for optimizing FNN that in turn address its local minima and generalization problems.

Initially, only gradient based linear approximation and quadratic approximation methods for optimizing FNNs were employed to train FNN. These conventional algorithms (backpropagation, Quickpro, Rprop, conjugate gradient, etc.) are local search algorithms that exploit current solution to generate a new solution; however, they lack in exploration ability, therefore, usually, finds local minima of an optimization problem.

Unlike conventional approaches, metaheuristics (e.g., genetic algorithm, particle swarm optimization, ant colony optimization, etc.) are good at both exploitation and exploration and can address simultaneous adaptation in each component of FNN. However, no single method can solve all kinds of problem. So, we need to improvise, adapt, and construct hybrid methods for optimizing FNN. Therefore, several dynamic designs of FNN are reported in the literature: EPNet (an adaptive method of FNN architecture optimization), neuro-evolution of augmenting topologies, flexible neural tree, cooperative coevolution neural network, etc., are among them. Hence, there is a wide spectrum of FNN optimization/adaptation is possible with metaheuristic treatment to FNNs (Fig. 3) in which the fundamental aspect is the formulation of FNN (phenotype) to vectored form (genotype) or any other form of mechanism for manipulation of FNN components.

Since there are many components to be manipulated by means of metaheuristic strategies and the availability of the fact that FNN generalization ability depends on the optimization its all the components, multiobjective treatment to FNN were used. The multiobjective-based training allows an FNN to evolve with handling two or more FNN-related objectives, such as approximation error, network complexity, input dimension, etc. Moreover, the generalization ability of system can be easily improved by combining decision of many candidates of the system. Hence, an ensemble of FNNs by making use of the metaheuristic final population was proposed and the two crucial aspect accuracy and diversity of an ensemble were taken care during propose of evolving FNNs.

It is evident from such aspects of FNN optimization that the future research will be able to bring the new paradigms of FNNs by applying or by the inspiration from the discussed methods in this article. Hence, that will overcome the data quality problem and will be handling new challenges of big data to cope-up with the new era information processing.

Acknowledgment

Authors would like to thank the all the anonymous reviewers for the technical comments, which enhanced the contents of the preliminary version of this paper. This work was supported by the IPROCOM Marie Curie Initial Training Network, funded through the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007–2013/, under REA grant agreement number 316555.

References

Abbass, 2003. Speeding up backpropagation using multiobjective evolutionary algorithms. Neural Comput. 15 (November (11)), 2705–2726.
Abbass, H.A., 2002. The self-adaptive pareto differential evolution algorithm. In: Proceedings of the 2002 Congress Evolutionary Computation, 2002. CEC '02 1, pp. 831–836.
Abraham, A., 2004. Meta learning evolutionary artificial neural networks. Neurocomputing 56 (January (0)), 1–38.
Ackley, D.H., Hinton, G.E., Sejnowski, T.J., 1985. A learning algorithm for boltzmann machines. Cogn. Sci. 9 (1), 147–169.
Akhand, M., Islam, M.M., Murase, K., 2009. Progressive interactive training: a sequential neural network ensemble learning method. Neurocomputing 73 (December (1)), 260–273.
Alba, E., Marti, R., 2006. Metaheuristic Procedures for Training Neural Networks. Springer.
de Albuquerque Teixeira, R., Braga, A.P., Takahashi, R.H., Saldanha, R.R., 2000. Improving generalization of MLPs with multi-objective optimization. Neurocomputing 35 (November (1)), 189–194.
Ali Ahmadi, M., Zendehboudi, S., Lohi, A., Elkamel, A., Chatzis, I., 2013. Reservoir permeability prediction by neural networks combined with hybrid genetic algorithm and particle swarm optimization. Geophys. Prospect. 61 (May (3)), 582–598.
Al-kazemi, B., Mohan, C., 2002. Training feedforward neural networks using multi-phase particle swarm optimization. In: Proceedings of the 9th International Conference Neural Information Processing. 2002, ICONIP '02, vol. 5, pp. 2615–2619.
Almeida, L.M., Ludermir, T.B., 2010. A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks. Neurocomputing 73 (March (7)), 1438–1450.
Alvarez, A., 2002. A neural network with evolutionary neurons. Neural Process. Lett. 16 (August (1)), 43–52.
Amari, S.-i., Murata, N., Muller, K.-R., Finke, M., Yang, H.H., 1997. Asymptotic statistical theory of overtraining and cross-validation. IEEE Trans. Neural Netw. 8 (September (5)), 985–996.
Andersen, H.C., Tsoi, A.C., 1993. A constructive algorithm for the training of a multilayer perceptron based on the genetic algorithm. Complex Syst. 7 (4), 249–268.
Andoni, A., Panigrahy, R., Valiant, G., Zhang, L., 2014. Learning polynomials with neural networks. In: Proceedings of the 31st International Conference on Machine Learning

(ICML-14), pp. 1908–1916.
Angeline, P.J., Saunders, G.M., Pollack, J.B., 1994. An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans. Neural Netw. 5 (1), 54–65.
Arifovic, J., Gencay, R., 2001. Using genetic algorithms to select architecture of a feedforward artificial neural network. Physica A 289 (January), 574–594.
Augusteijn, M.F., Harrington, T.P., 2004. Evolving transfer functions for artificial neural networks. Neural Comput. Appl. 13 (April (1)), 38–46.
Azimi-Sadjadi, M.R., Liou, R.-J., 1992. Fast learning process of multilayer neural networks using recursive least squares method. IEEE Trans. Signal Process. 40 (2), 446–450.
Bakker, B., Heskes, T., 2003. Clustering ensembles of neural network models. Neural Netw. 16 (March (2)), 261–269.
Baranyi, J., Pin, C., Ross, T., 1999. Validating and comparing predictive models. Int. J. Food Microbiol. 48 (3), 159–166.
Barnard, E., Cole, R.A., 1989. A neural-net training program based on conjugate-gradient optimization. Technical Repport CSE 89-014, Department of Computer Science, Oregon Graduate Institute of Science and Technology, Tech. Rep.
Battiti, R., Tecchiolli, G., 1995. Training neural nets with the reactive tabu search. IEEE Trans. Neural Netw. 6 (5), 1185–1200.
Baxter, J., 1992. The evolution of learning algorithms for artificial neural networks. In: Complex Syst., pp. 313–326.
Behrman, E.C., Niemel, J., Steck, J.E., Skinner, S.R., 1996. A quantum dot neural network. In: Proceedings of the 4th Workshop on Physics of Computation, pp. 22–24.
Belew, R.K., Mcinerney, J., Schraudolph, N.N., 1990. Evolving networks: using the genetic algorithm with connectionist learning. University of California, San Diego, Tech. Rep. CS90-174.
Bertsekas, D.P., 1999. Nonlinear Programming 2nd ed.. Athena scientific Belmont.
Beyer, D., Ogier, R., 1991. Tabu learning: a neural network search method for solving nonconvex optimization problems. In: Proceedings of the International Jt. Conference Neural Networks, 1991. IJCNN, vol. 2, pp. 953–961.
Bishop, C.M., 1995. Training with noise is equivalent to tikhonov regularization. Neural Comput. 7 (1), 108–116.
Bland, R.G., Goldfarb, D., Todd, M.J., 1981. The ellipsoid method: a survey. Oper. Res. 29 (6), 1039–1091.
Bouaziz, S., Alimi, A.M., Abraham, A., 2014. Universal approximation propriety of flexible beta basis function neural tree. In: Proceedings of the International Jt. Conference Neural Networking (IJCNN), pp. 573–580.
Bousquet, O., Elisseeff, A., 2002. Stability and generalization. J. Mach. Learn. Res. 2, 499–526.
Boussaid, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Inform. Sci. 237 (July), 82–117.
Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (August (2)), 123–140.
Brownlee, J., 2011. Clever Algorithms: Nature-inspired Programming Recipes. Jason Brownlee.
Carvalho, A.R., Ramos, F.M., Chaves, A.A., 2011. Metaheuristics for the feedforward artificial neural network (ann) architecture optimization problem. Neural Comput. Appl. 20 (December (8)), 1273–1284.
Carvalho, M., Ludermir, T., 2007. Particle swarm optimization of neural network architectures andweights. In: Proceedings of the 7th International Conference Hybrid Intelligent Systems, HIS, pp. 336–339.
Černỳ, V., 1985. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J. Optim. Theory Appl. 45 (January (1)), 41–51.
Cetin, B.C., Burdick, J.W., Barhen, J., 1993. Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks. In: Proceedings of the IEEE International Conference Neural Networks, pp. 836–842.
Chalmers, D.J., 1990. The evolution of learning: an experiment in genetic connectionism. In: Proceedings of the 1990 Connectionist Models Summer School, pp. 81–90.
Chandra, A., Yao, X., 2006. Ensemble learning using multi-objective evolutionary algorithms. J. Math. Model. Algorithms 5 (December (4)), 417–445.
Charalambous, C., 1992. Conjugate gradient algorithm for efficient training of artificial neural networks. IEE Proc. G (Circuits, Devices, Syst.) 139 (June (3)), 301–310.
Chen, H., Yao, X., 2010. Multiobjective neural network ensembles based on regularized negative correlation learning. IEEE Trans. Knowl. Data Eng. 22 (February (12)), 1738–1751.
Chen, L., Xue, W., Tokuda, N., 2010. Classification of 2-dimensional array patterns: assembling many small neural networks is better than using a large one. Neural Netw. 23 (August (6)), 770–781.
Chen, Y., Yang, B., Dong, J., 2004. Nonlinear system modelling via optimal design of neural trees. Int. J. Neural Syst. 14 (April (2)), 125–137.
Chen, Y., Abraham, A., Yang, B., 2006. Feature selection and classification using flexible neural tree. Neurocomputing 70 (December (1)), 305–313.
Chen, O.T.-C., Sheu, B.J., 1994. Optimization schemes for neural network training. In: Proceedings of the IEEE International Conference Neural Networks and IEEE World Congess Computational Intelligence, vol. 2, pp. 817–822.
Cho, S., Jang, M., Chang, S., 1997. Virtual sample generation using a population of networks. Neural Process. Lett. 5 (2), 21–27.
Cho, S., Cha, K., 1996. Evolution of neural network training set through addition of virtual samples. In: Proceedings of the IEEE International Conference Evolutionary Computation, pp. 685–688.
Chrisley, R.L., 1997. Learning in Non-superpositional Quantum Neurocomputers. Brain, Mind and Physics. IOS Press, Amsterdam, 126–139.
Coello, C.A.C., 1999. A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl. Inform. Syst. 1 (August (3)), 129–156.
Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20 (September (3)), 273–297.
Costa, M.A., Braga, A.P., Menezes, B.R., Teixeira, R.A., Parma, G.G., 2003. Training neural networks with a multi-objective sliding mode control algorithm. Neurocomputing 51 (April), 467–473.
Cruz-Ramírez, M., Sánchez-Monedero, J., Fernández-Navarro, F., Fernández, J., Hervás-Martínez, C., 2010. Memetic pareto differential evolutionary artificial neural networks to determine growth multi-classes in predictive microbiology. Evol. Intell. 3 (December (3–4)), 187–199.
Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. 2 (December (4)), 303–314.
Da, Y., Xiurun, G., 2005. An improved PSO-based ANN with simulated annealing technique. Neurocomputing 63 (January (0)), 527–533.
Dai, Y.-H., Yuan, Y., 1999. A nonlinear conjugate gradient method with a strong global convergence property. SIAM J. Optim. 10 (1), 177–182.
Das, I., Dennis, J.E., 1997. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Struct. Optim. 14 (1), 63–69.
Das, S., Suganthan, P.N., 2011. Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15 (October (1)), 4–31.
Dasgupta, D., McGregor, D., 1992. Designing application-specific neural networks using the structured genetic algorithm. In: Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992, COGANN-92, pp. 87–96.
Davis, M.H., Vinter, R.B., 1985. Stochastic Modelling and Control. Chapman and Hall, London, UK.
Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., Schwefel, H.-P. (Eds.), Parallel Problem Solving from Nature PPSN VI, ser. Lecture Notes in Computer Science, vol. 1917, Springer, pp. 849–858.
Deneubourg, J.-L., Aron, S., Goss, S., 1990. The self-organizing exploratory pattern of the argentine ant. J. Insect Behav. 3 (March), 159–169.
Dhahri, H., Alimi, A., Abraham, A., 2013. Hierarchical particle swarm optimization for the design of beta basis function neural network. In: Abraham, A., Thampi, S.M. (Eds.), Intelligent Informatics, ser. Advances in Intelligent Systems and Computing 182. Springer, 193–205.
Diebold, F.X., Mariano, R.S., 1995. Comparing predictive accuracy. J. Bus. Econ. Stat. 13 (3), 253–263.
Ding, S., Su, C., Yu, J., 2011. An optimizing bp neural network algorithm based on genetic algorithm. Artif. Intell. Rev. 36 (August (2)), 153–162.
Ditzler, G., Roveri, M., Alippi, C., Polikar, R., 2015. Learning in nonstationary environments: a survey. IEEE Comput. Intell. Mag. 10 (4), 12–25.
Dominey, P.F., 1995. Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biol. Cybern. 73 (August (3)), 265–274.
Donate, J.P., Li, X., Sánchez, G.G., de Miguel, A.S., 2013. Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. Neural Comput. Appl. 22 (January (1)), 11–20.
Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. 26 (February (1)), 29–41.
Dumont, J.P., Robertson, R.M., et al., 1986. Neuronal circuits: an evolutionary perspective. Science 233 (August (4766)), 849–853.
Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, 1995. MHS '95, pp. 39–43.
Engel, J., 1988. Teaching feed-forward neural networks by simulated annealing. Complex Syst. 2 (6), 641–648.
Fahlman, S.E., Lebiere, C., 1990. The cascade-correlation learning architecture. In: Touretzky, D.S. (Ed.), Advances in Neural Information Processing Systems 2. Morgan Kaufmann, San Francisco, CA, USA, 524–532.
Fahlman, S.E., 1988. An empirical study of learning speed in back-propagation networks. Carnegie Mellon University, Tech. Rep.
Feo, T.A., Resende, M.G., 1995. Greedy randomized adaptive search procedures. J. Glob. Optim. 6 (June (2)), 109–133.
FernandezCaballero, J., Martinez, F., Hervas, C., Gutierrez, P., 2010. Sensitivity versus accuracy in multiclass problems using memetic pareto evolutionary neural networks. IEEE Trans. Neural Netw. 21 (May (5)), 750–770.
Fister, I., Jr., Yang, X.-S., Fister, I., Brest, J., Fister, D., 2013. A brief review of nature-inspired algorithms for optimization. Elektro. Vestn. (Engl. Ed.) 80 (July (3)).
Fletcher, R., 1987. Practical Methods of Optimization. John Wiley & Sons.
Fodor, I.K., 2002. A survey of dimension reduction techniques. Lawrence Livermore National Laboratory, Tech. Rep. UCRL-ID-148494.
Fogel, D.B., 1998. Evolutionary Computation: The Fossil Record. Wiley-IEEE Press.
Fogel, D.B., Fogel, L.J., Porto, V., 1990. Evolving neural networks. Biol. Cybern. 63 (October (6)), 487–493.
Fontanari, J., Meir, R., 1991. Evolving a learning algorithm for the binary perceptron. Netw.: Comput. Neural Syst. 2 (4), 353–359.
Formato, R.A., 2007. Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog. Electromagn. Res. 77 (January), 425–491.
Frean, M., 1990. The UPSTART algorithm: a method for constructing and training feedforward neural networks. Neural Comput. 2 (April (2)), 198–209.
Fukumizu, K., Amari, S.-I., 2000. Local minima and plateaus in hierarchical structures of multilayer perceptrons. Neural Netw. 13 (May (3)), 317–327.
Fullér, R., 2013. Introduction to Neuro-fuzzy Systems 2. Springer Science & Business Media.
Furtuna, R., Curteanu, S., Leon, F., 2011. An elitist non-dominated sorting genetic algorithm enhanced with a neural network applied to the multi-objective optimization of a polysiloxane synthesis process. Eng. Appl. Artif. Intell. 24 (August

(5)), 772–785.
Garcia-Pedrajas, N., Hervas-Martinez, C., Munoz-Perez, J., 2003. COVNET: a cooperative coevolutionary model for evolving artificial neural networks. IEEE Trans. Neural Netw. 14 (May (3)), 575–596.
García-Pedrajas, N., Hervás-Martínez, C., Muñoz Pérez, J., 2002a. Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). Neural Netw. 15 (December (10)), 1259–1278.
García-Pedrajas, N., Ortiz-Boyer, D., Hervás-Martínez, C., 2006. An alternative approach for neural network evolution with a genetic algorithm: crossover by combinatorial optimization. Neural Netw. 19 (May (4)), 514–528.
García-Pedrajas, N., Hervás-Martínez, C., Muñoz Pérez, J., 2002. SYMBIONT: a cooperative evolutionary model for evolving artificial neural networks for classification. In: Bouchon-Meunier, B., Gutiérrez-Ríos, J., Magdalena, L., Yager, R. R. (Eds.), Technologies for Constructing Intelligent Systems 2, ser. Studies in Fuzziness and Soft Computing, vol. 90, Physica-Verlag HD, 2002, pp. 341–354.
Garro, B.A., Sossa, H., Vázquez, R.A., 2011. Artificial neural network synthesis by means of artificial bee colony (ABC) algorithm. In: Proceedings of the IEEE Congress Evolutionary Compuation (CEC), pp. 331–338.
Gaspar-Cunha, A., Vieira, A., 2005. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. Int. J. Comput. Syst. Signals 6 (January (1)), 18–36.
Gauci, J., Stanley, K., 2007. Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the 9th Annual Conference Genetic Evolutionary Computation, ACM, pp. 997–1004.
Geem, Z.W., Kim, J.H., Loganathan, G., 2001. A new heuristic optimization algorithm: harmony search. Simulation 76 (February (2)), 60–68.
Geman, S., Bienenstock, E., Doursat, R., 1992. Neural networks and the bias/variance dilemma. Neural Comput. 4 (January (1)), 1–58.
Gershenfeld, N., Chuang, I.L., 1998. Quantum computing with molecules. Sci. Am. 278 (6), 66–71.
Ghalambaz, M., Noghrehabadi, A., Behrang, M., Assareh, E., Ghanbarzadeh, A., Hedayat, N., 2011. A hybrid neural network and gravitational search algorithm (HNNGSA) method to solve well known wessinger's equation. World Acad. Sci. Eng. Technol. 5 (January), 803–807.
Girosi, F., Jones, M., Poggio, T., 1995. Regularization theory and neural networks architectures. Neural Comput. 7 (March (2)), 219–269.
Giustolisi, O., Simeone, V., 2006. Optimal design of artificial neural networks by a multi-objective strategy: groundwater level predictions. Hydrol. Sci. J. 51 (January (3)), 502–523.
Glover, F., 1989. Tabu search-part I. INFORMS J. Comput. 1 (3).
Goh, C.-K., Teoh, E.-J., Chen Tan, K., 2008. Hybrid multiobjective evolutionary design for artificial neural networks. IEEE Trans. Neural Netw. 19 (July (9)), 1531–1548.
Goldberg, D.E., Holland, J.H., 1988. Genetic algorithms and machine learning. Mach. Learn. 3 (October (2)), 95–99.
Gori, M., Tesi, A., 1992. On the problem of local minima in backpropagation. IEEE Trans. Pattern Anal. Mach. Intell. 14 (January (1)), 76–86.
Gorin, A., Mammone, R., 1994. Introduction to the special issue on neural networks for speech processing. IEEE Trans. Speech Audio Process. 2 (January (1)), 113–114.
Green, R.C., II, Wang, L., Alam, M., 2012. Training neural networks using central force optimization and particle swarm optimization: insights and comparisons. Expert Syst. Appl. 39 (December (1)), 555–563.
Grossberg, S., 1987. Competitive learning: from interactive activation to adaptive resonance. Cogn. Sci. 11 (January (1)), 23–63.
Guo, Z., Uhrig, R.E., 1992. Using genetic algorithms to select inputs for neural networks. In: Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992. COGANN-92, pp. 223–234.
Hagan, M.T., Menhaj, M.B., 1994. Training feedforward networks with the Marquardt algorithm. IEEE Trans. Neural Netw. 5 (November (6)), 989–993.
Hansen, L., Salamon, P., 1990. Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. 12 (October (10)), 993–1001.
Harp, S.A., Samad, T., Guha, A., 1989. Towards the genetic synthesis of neural network. In: Proceedings of the 3rd International Conference Genetic Algorithms, pp. 360–369.
Haykin, S.S., 2001. Kalman Filtering and Neural Networks. Wiley Online Library.
Haykin, S.S., 2009. Neural Networks and Learning Machines 3. Pearson Education Upper Saddle River.
Hernández, M.A., Stolfo, S.J., 1998. Real-world data is dirty: data cleansing and the merge/purge problem. Data Min. Knowl. Discov. 2 (1), 9–37.
Hestenes, M.R., Stiefel, E., 1952. Methods of conjugate gradients for solving linear systems. J. Res. Nat. Bur. Stand. 49 (December (6)), 409–436.
Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. 29 (November (6)), 82–97.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.
Hinton, G.E., Osindero, S., Teh, Y.-W., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18 (July (7)), 1527–1554.
Hirose, A., 2006. Complex-valued Neural Networks. Springer Science & Business Media.
Ho, Y.-C., Pepyne, D.L., 2002. Simple explanation of the no free lunch theorem of optimization. Cybern. Syst. Anal. 38 (March (2)), 292–298.
Holland, J.H., 1975. Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press.
Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. USA 79 (April (8)), 2554–2558.
Horng, M.-H., Lee, M.-C., Liou, R.-J., Lee, Y.-X., 2012. Firefly meta-heuristic algorithm for training the radial basis function network for data classification and disease diagnosis. In: Parpinelli, R., Lopes, H.S. (Eds.), Theory and New Applications of Swarm Intelligence, InTech.
Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural Netw. 4 (October (2)), 251–257.
Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2 (March (5)), 359–366.
Huang, D.-S., 1998. The local minima-free condition of feedforward neural networks for outer-supervised learning. IEEE Trans. Syst. Man Cybern. B Cybern. 28 (June (3)), 477–480.
Huang, G.-B., 2014. An insight into extreme learning machines: random neurons, random features and kernels. Cogn. Comput. 6 (3), 376–390.
Huang, G.-B., Babri, H.A., 1998. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans. Neural Netw. 9 (January (1)), 224–229.
Huang, G.-B., Chen, L., Siew, C.-K., 2006a. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Netw. 17 (July (4)), 879–892.
Huang, G.-B., Zhu, Q.-Y., Siew, C.-K., 2006b. Extreme learning machine: theory and applications. Neurocomputing 70 (1), 489–501.
Igel, C., Toussaint, M., 2003. On classes of functions for which no free lunch results hold. Inf. Process. Lett. 86 (6), 317–321.
Ilonen, J., Kamarainen, J.-K., Lampinen, J., 2003. Differential evolution training algorithm for feed-forward neural networks. Neural Process. Lett. 17 (February (1)), 93–105.
Irani, R., Nasimi, R., 2011. Evolving neural network using real coded genetic algorithm for permeability estimation of the reservoir. Expert Syst. Appl. 38 (August (8)), 9862–9866.
Irani, R., Nasimi, R., 2012. An evolving neural network using an ant colony algorithm for a permeability estimation of the reservoir. Pet. Sci. Technol. 30 (February (4)), 375–384.
Islam, M.M., Yao, X., Murase, K., 2003. A constructive algorithm for training cooperative neural network ensembles. IEEE Trans. Neural Netw. 14 (July (4)), 820–834.
Ismail, A., Engelbrecht, A., 2000. Global optimization algorithms for training product unit neural networks. In: Proceedings of the IEEE-INNS-ENNS International Jt. Conference Neural Networks. IJCNN, vol. 1, pp. 132–137.
Jacobs, R.A., 1988. Increased rates of convergence through learning rate adaptation. Neural Netw. 1 (4), 295–307.
Jaeger, H., 2001. The echo state approach to analysing and training recurrent neural networks-with an erratum note. German National Research Center for Information Technology, Bonn, Germany, Tech. Rep.
Jain, A., Duin, R., Mao, J., 2000. Statistical pattern recognition: a review. IEEE Trans. Pattern Anal. Mach. Intell. 22 (January (1)), 4–37.
Jain, A.K., Mao, J., Mohiuddin, K.M., 1996. Artificial neural networks: a tutorial. Comput 29 (March (3)), 31–44.
Jin, Y., Sendhoff, B., 2008. Pareto-based multiobjective machine learning: an overview and case studies. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 38 (3), 397–415.
Jin, Y., Okabe, T., Sendhoff, B., 2004. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In: Proceedings of the Congress Evolutionary Computation, 2004. CEC2004 1, pp. 1–8.
Jin, Y., Sendhoff, B., Körner, E., 2005. Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations. In: Evol. Multi-Criterion Optim., ser. Lecture Notes in Computer Science, vol. 3410, 2005, pp. 752–766.
Jin, Y., Sendhoff, B., Körner, E., 2005. Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations. In: Evolutionary Multi-Criterion Optimization, ser. Lecture Notes in Computer Science, vol. 3410. Springer, pp. 752–766.
Juang, C.-F., 2004. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. Syst., Man, Cybern. B, Cybern. 34 (April (2)), 997–1006.
Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement learning: a survey. J. Artif. Intell. Res. 4 (May), 237–285.
Karaboga, D., Akay, B., Ozturk, C., 2007. Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In: Torra, V., Narukawa, Y., Yoshida, Y. (Eds.), Modeling Decisions for Artificial Intelligence, ser. Lecture Notes in Computer Science 4617. Springer, 318–329.
Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization. Computer Engineering Department, Erciyes University, Technical Report TR06.
Karpat, Y., Özel, T., 2007. Multi-objective optimization for turning processes using neural network modeling and dynamic-neighborhood particle swarm optimization. Int. J. Adv. Manuf. Technol. 35 (December (3–4)), 234–247.
Kassahun, Y., Sommer, G., 2005. Efficient reinforcement learning through evolutionary acquisition of neural topologies. In: Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN 2005), pp. 259–266.
Kattan, A., Abdullah, R., Salam, R., 2010. Harmony search based supervised training of artificial neural networks. In: Proceedings of the 2010 International Conference International Systems, Model. and Simulation (ISMS), pp. 105–110.
Kennedy, J.F., Kennedy, J., Eberhart, R.C., 2001. Swarm Intelligence. Morgan Kaufmann.
Kennedy, J., Eberhart, R.C., 1997. A discrete binary version of the particle swarm algorithm. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetic on Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108.
Khan, K., Sahai, A., 2012. A comparison of BA, GA, PSO, BP and LM for training feed

forward neural networks in e-learning context. Int. J. Intell. Syst. Appl. 4 (7), 23.
Khan, M.M., Ahmad, A.M., Khan, G.M., Miller, J.F., 2013. Fast learning neural networks using cartesian genetic programming. Neurocomputing 121 (December), 274–289.
Kim, D., Kim, H., Chung, D., 2005. A modified genetic algorithm for fast training neural networks. In: Wang, J., Liao, X., Yi, Z. (Eds.), Advances in Neural Networks – ISNN 2005, ser. Lecture Notes in Computer Science 3496. Springer, 660–665.
Kim, E., Helal, S., Cook, D., 2010. Human activity recognition and pattern discovery. IEEE Pervasive Comput. 9 (1), 48–53.
Kim, H.B., Jung, S.H., Kim, T.G., Park, K.H., 1996. Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates. Neurocomputing 11 (May (1)), 101–106.
Kim, K.-J., Cho, S.-B., 2008. Evolutionary ensemble of diverse artificial neural networks using speciation. Neurocomputing 71 (March (7)), 1604–1618.
Kiranyaz, S., Ince, T., Yildirim, A., Gabbouj, M., 2009. Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. Neural Netw. 22 (December (10)), 1448–1462.
Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Sci 220 (May (4598)), 671–680.
Kitano, H., 1990a. Designing neural networks using genetic algorithms with graph generation system. Complex Syst. 4 (4), 461–476.
Kitano, H., 1994. Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms. Physica D 75 (August (1)), 225–238.
Kitano, H., 1990. Empirical studies on the speed of convergence of neural network training using genetic algorithms. In: Proceedings of the 8th National Conference Artificial Intelligence, vol. 2, pp. 789–795.
Kohonen, T., 1982. Self-organized formation of topologically correct feature maps. Biol. Cybern. 43 (January (1)), 59–69.
Kolmogorov, A.K., 1957. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. Dokl. Akad. Nauk SSSR 114, 369–373, (in Russian).
Kordík, P., Koutník, J., Drchal, J., Kovářík, O., Čepek, M., Šnorek, M., 2010. Meta-learning approach to neural network optimization. Neural Netw. 23 (May (4)), 568–582.
Kouda, N., Matsui, N., Nishimura, H., Peper, F., 2005. Qubit neural network and its learning efficiency. Neural Comput. Appl. 14 (2), 114–121.
Koza, J.R., Bennett, F.H., III, Stiffelman, O., 1999. Genetic Programming as a Darwinian Invention Machine. Springer.
Koza, J.R., Rice, J.P., 1991. Genetic generation of both the weights and architecture for a neural network. In: Proceedings of the International Jt. Conference Neural Networks, IJCNN, vol. 2, pp. 397–404.
Kulluk, S., Ozbakir, L., Baykasoglu, A., 2012. Training neural networks with harmony search algorithms for classification problems. Eng. Appl. Artif. Intell. 25 (February (1)), 11–19.
Kůrková, V., 1992. Kolmogorov's theorem and multilayer neural networks. Neural Netw. 5 (3), 501–506.
Lam, H., Leung, F., 2006. Design and stabilization of sampled-data neural-network-based control systems. IEEE Trans. Syst. Man Cybern. B, Cybern. 36 (October (5)), 995–1005.
Larrañaga, P., Lozano, J.A., 2002. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation 2. Springer.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.
Lera, G., Pinzolas, M., 2002. Neighborhood based levenberg-marquardt algorithm for neural network training. IEEE Trans. Neural Netw. 13 (September (5)), 1200–1203.
Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S., 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw. 6 (March (6)), 861–867.
Leung, F.H.F., Lam, H., Ling, S., Tam, P.-S., 2003. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Trans. Neural Netw. 14 (January (1)), 79–88.
Leung, Y., Gao, Y., Xu, Z.-B., 1997. Degree of population diversity-a perspective on premature convergence in genetic algorithms and its markov chain analysis. IEEE Trans. Neural Netw. 8 (5), 1165–1176.
Lewenstein, M., 1994. Quantum perceptrons. J. Mod. Opt. 41 (12), 2491–2501.
Li, P., Xiao, H., Shang, F., Tong, X., Li, X., Cao, M., 2013. A hybrid quantum-inspired neural networks with sequence inputs. Neurocomputing 117, 81–90.
Lin, C.-J., Chen, C.-H., Lin, C.-T., 2009. A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications. IEEE Trans. Syst. Man, Cybern. C, Appl. Rev. 39 (December (1)), 55–68.
Lin, S.-W., Ying, K.-C., Chen, S.-C., Lee, Z.-J., 2008. Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst. Appl. 35 (4), 1817–1824.
Ling, S., Leung, F., Lam, H., 2007. Input-dependent neural network trained by real-coded genetic algorithm and its industrial applications. Soft Comput. 11 (September (11)), 1033–1052.
Lippmann, R.P., 1987. An introduction to computing with neural nets. IEEE ASSP Mag. 4 (April (2)), 4–22.
Liu, Y., Yao, X., 1999. Ensemble learning via negative correlation. Neural Netw. 12 (December (10)), 1399–1404.
Liu, Y., Yao, X., Higuchi, T., 2000. Evolutionary ensembles with negative correlation learning. IEEE Trans. Evol. Comput. 4 (November (4)), 380–387.
Liu, Y., Yao, X., 1996. Evolutionary design of artificial neural networks with different nodes. In: Proceedings of the IEEE International Conference Evolutionary Computation, pp. 670–675.
Lowe, D., Broomhead, D., 1988. Multivariable functional interpolation and adaptive networks. Complex Syst. 2, 321–355.
Ludermir, T., Yamazaki, A., Zanchettin, C., 2006. An optimization methodology for neural network weights and architectures. IEEE Trans. Neural Netw. 17 (November (6)), 1452–1459.
Mahdavi, M., Fesanghary, M., Damangir, E., 2007. An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. 188 (May (2)), 1567–1579.
Mani, G., 1990. Learning by gradient descent in function space. In: Proceedings of the IEEE International Conference on Systems, Man, Cybern., pp. 242–247.
Maniezzo, V., 1994. Genetic evolution of the topology and weight distribution of neural networks. IEEE Trans. Neural Netw. 5 (January (1)), 39–53.
March, J.G., 1991. Exploration and exploitation in organizational learning. Org. Sci. 2 (February (1)), 71–87.
Marquardt, D.W., 1963. An algorithm for least-squares estimation of nonlinear parameters. J. Ind. Appl. Math. 11 (June (2)), 431–441.
Martínez-Muñoz, G., Sánchez-Martínez, A., Hernández-Lobato, D., Suárez, A., 2008. Class-switching neural network ensembles. Neurocomputing 71 (August (13)), 2521–2528.
Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 922–928.
Mazurowski, M.A., Habas, P.A., Zurada, J.M., Lo, J.Y., Baker, J.A., Tourassi, G.D., 2008. Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance. Neural Netw. 21 (2), 427–436.
McCulloch, W., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biol. 5 (4), 115–133.
Menczer, F., Parisi, D., 1992. Evidence of hyperplanes in the genetic learning of neural networks. Biol. Cybern. 66 (January (3)), 283–289.
Menneer, T., Narayanan, A., 1995. Quantum-inspired neural networks. University of Exeter, Technical Report. R329.
Merrill, J.W., Port, R.F., 1991. Fractally configured neural networks. Neural Netw. 4 (1), 53–60.
Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. J. Chem. Phys. 21 (6), 1087–1092.
Minku, F.L., Ludermir, T.B., 2008. Clustering and co-evolution to construct neural network ensembles: an experimental study. Neural Netw. 21 (November (9)), 1363–1379.
Minsky, M., Papert, S.A., 1988. Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge, Mass.
Mirjalili, S., Mohd Hashim, S.Z., Moradian Sardroudi, H., 2012. Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl. Math. Comput. 218 (July (22)), 11 125–11 137.
Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.
Mitra, S., Hayashi, Y., 2006. Bioinformatics with soft computing. IEEE Trans. Syst. Man Cybern. C, Appl. Rev. 36 (September (5)), 616–635.
Mjolsness, E., Sharp, D.H., Alpert, B.K., 1989. Scaling, machine learning, and genetic neural nets. Adv. Appl. Math. 10 (June (2)), 137–163.
Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (November (11)), 1097–1100.
Montana, D.J., Davis, L., 1989. Training feedforward neural networks using genetic algorithms. In: Proceedings of the 11th International Jt. Conference Artificial Intelligence, vol. 1, pp. 762–767.
Moriarty, D.E., Miikkulainen, R., 1997. Forming neural networks through efficient and adaptive coevolution. Evol. Comput. 5 (December (4)), 373–399.
Moscato, P., 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. In: Proceedings of the Caltech, Technical Report Caltech Concurrent Computation Program, C3P Report.
Murray, A.F., Edwards, P.J., 1994. Enhanced mlp performance and fault tolerance resulting from synaptic weight noise during training. IEEE Trans. Neural Netw. 5 (5), 792–802.
Nakama, T., 2009. Theoretical analysis of batch and on-line training for gradient descent learning in neural networks. Neurocomputing 73 (1), 151–159.
Nandy, S., Sarkar, P.P., Das, A., 2012. Analysis of a nature inspired firefly algorithm based back-propagation neural network training. Int. J. Comput. Appl. 43 (April (2)), 8–16.
Narayanan, A., Menneer, T., 2000. Quantum artificial neural network architectures and components. Inf. Sci. 128 (3), 231–255.
Natschläger, T., Maass, W., Markram, H., 2002. The "liquid computer": a novel strategy for real-time computing on time series. Spec. Issue Found. Inf. Process. Telemat. 8 (1), 39–43.
Nedjah, N., Abraham, A., Mourelle, L.M., 2007. Hybrid artificial neural network. Neural Comput. Appl. 16 (May (3)), 207–208.
Niranjan, M., Principe, J., 1997. The past, present, and future of neural networks for signal processing. IEEE Signal Process. Mag. 14 (November (6)), 28–48.
Niu, B., Zhu, Y., He, X., Wu, H., 2007. MCPSO: a multi-swarm cooperative particle swarm optimizer. Appl. Math. Comput. 185 (February (2)), 1050–1062.
Nolfi, S., Parisi, D., Elman, J.L., 1994. Learning and evolution in neural networks. Adapt. Behav. 3 (June (1)), 5–28.
Oh, S.-K., Pedrycz, W., 2002. The design of self-organizing polynomial neural networks. Inf. Sci. 141 (3), 237–258.
Ojha, V.K., Abraham, A., Snášel, V., 2017. Ensemble of heterogeneous flexible neural trees using multiobjective genetic programming. Appl. Soft Comput. 52 (March), 909–924.
Ojha, V.K., Abraham, A., Snášel, V., December 2014. Simultaneous optimization of neural network weights and active nodes using metaheuristics. In: Proceedings of the

14th International Conference on Hybrid Intellectual System (HIS), pp. 248–253.
Osman, I.H., Laporte, G., 1996. Metaheuristics: a bibliography. Ann. Oper. Res. 63 (October (5)), 511–623.
Ozturk, C., Karaboga, D., 2011. Hybrid artificial bee colony algorithm for neural network training. In: Proceedings of the IEEE Congress Computational Intelligence (CEC), 2011, pp. 84–88.
Pan, Q.-K., Suganthan, P., Tasgetiren, M.F., Liang, J., 2010. A self-adaptive global best harmony search algorithm for continuous optimization problems. Appl. Math. Comput. 216 (April (3)), 830–848.
Passino, K.M., 2002. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. 22 (January (3)), 52–67.
Pavlidis, P., Weston, J., Cai, J., Grundy, W.N., 2001. Gene functional classification from heterogeneous data. In: Proceedings of the 5th Annual International Conference on Computational Biology ACM, pp. 249–255.
Pearce, J., Ferrier, S., 2000. Evaluating the predictive performance of habitat models developed using logistic regression. Ecol. Modell. 133 (3), 225–245.
Pencina, M.J., D Agostino, R.B., Vasan, R.S., 2008. Evaluating the added predictive ability of a new marker: from area under the roc curve to reclassification and beyond. Stat. Med. 27 (2), 157–172.
Peng, L., Yang, B., Zhang, L., Chen, Y., 2011. A parallel evolving algorithm for flexible neural tree. Parallel Comput. 37 (October (10–11)), 653–666.
Pettersson, F., Chakraborti, N., Saxén, H., 2007. A genetic algorithms based multi-objective neural net applied to noisy blast furnace data. Appl. Soft Comput. 7 (January (1)), 387–397.
Pipino, L.L., Lee, Y.W., Wang, R.Y., 2002. Data quality assessment. Commun. ACM 45 (April (4)), 211–218.
Polikar, R., 2006. Ensemble based systems in decision making. IEEE Circuits Syst. Mag. 6 (3), 21–45.
Poston, T., Lee, C.-N., Choie, Y., Kwon, Y., 1991. Local minima and back propagation. In: Proceedings of the International Jt. Conference Neural Networks, IJCNN. vol. 2, pp. 173–176.
Prechelt, L., 1998. Automatic early stopping using cross validation: quantifying the criteria. Neural Netw. 11 (4), 761–767.
Prisecaru, P., 2016. Challenges of the fourth industrial revolution. Knowl. Horiz. Econ. 8 (1), 57.
Puig, V., Witczak, M., Nejjari, F., Quevedo, J., Korbicz, J., 2007. A gmdh neural network-based approach to passive robust fault detection using a constraint satisfaction backward test. Eng. Appl. Artif. Intell. 20 (7), 886–897.
Qin, Z., Liu, Y., Heng, X., Wang, X., 2005. Negatively correlated neural network ensemble with multi-population particle swarm optimization. In: Wang, J., Liao, X., Yi, Z. (Eds.), Advances in Neural Networks-ISNN 2005, ser. Lecture Notes in Computer Science, vol. 3496, Springer, pp. 520–525.
Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009. Gsa: a gravitational search algorithm. Inform. Sci. 179 (June (13)), 2232–2248.
Reed, R., Marks, R.J., Oh, S., 1995. Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. IEEE Trans. Neural Netw. 6 (3), 529–538.
Riedmiller, M., Braun, H., 1993. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: IEEE International Conference Neural Networks, IJCNN. pp. 586–591.
Ritchie, M.D., Holzinger, E.R., Li, R., Pendergrass, S.A., Kim, D., 2015. Methods of integrating data to uncover genotype-phenotype interactions. Nat. Rev. Genet. 16 (2), 85–97.
Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. 65 (November (6)), 386–408.
Roth, S., Gepperth, A., Igel, C., 2006. Multi-objective neural network optimization for visual object detection. In: Jin, Y. (Ed.), Multi-Objective Machine Learning, ser. Studies in Computational Intelligence, vol. 16, Springer, pp. 629–655.
Rudolph, T.G., 2011. A heuristic review of quantum neural networks. Ph.D. dissertation, Imperial College London.
Rumelhart, D.E., Zipser, D., 1985. Feature discovery by competitive learning. Cogn. Sci. 9 (January (1)), 75–112.
Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323 (October).
Saad, D., 2009. On-line Learning in Neural Networks 17. Cambridge University Press.
Salajegheh, E., Gholizadeh, S., 2005. Optimum design of structures by an improved genetic algorithm using neural networks. Adv. Eng. Softw. 36 (November (11–12)), 757–767.
Sarangi, P.P., Sahu, A., Panda, M., 2014. Training a feed-forward neural network using artificial bee colony with back-propagation algorithm. In: Proceedings of the Intelligent Computing, Networking, and Informatics. Springer, pp. 511–519.
Sarkar, D., Modak, J.M., 2003. ANNSA: a hybrid artificial neural network/simulated annealing algorithm for optimal control problems. Chem. Eng. Sci. 58 (July (14)), 3131–3142.
Schaffer, J.D., Caruana, R.A., Eshelman, L.J., 1990. Using genetic search to exploit the emergent behavior of neural networks. Physica D 42 (June (1)), 244–248.
Schapire, R.E., 1990. The strength of weak learnability. Mach. Learn. 5 (June (2)), 197–227.
Schiffmann, W., Joost, M., Werner, R., 1994. Optimization of the backpropagation algorithm for training multilayer perceptrons. University of Koblenz, Institute of Physics, Rheinau, Koblenz, Tech. Rep.
Schmidhuber, J., 2015. Deep learning in neural networks: an overview. Neural Netw. 61, 85–117.
Schraudolph, N.N., Yu, J., Günter, S., et al., 2007. A stochastic quasi-newton method for online convex optimization. In: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, vol. 7, pp. 436–443.
Schumacher, C., Vose, M.D., Whitley, L.D., 2001. The no free lunch and problem description length. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 565–570.
Schwefel, H.-P., 1987. Collective Phenomena in Evolutionary Systems. Universität Dortmund. Abteilung Informatik.
Sejnowski, T.J., Rosenberg, C.R., 1987. Parallel networks that learn to pronounce english text. Complex Syst. 1 (1), 145–168.
Selmic, R., Lewis, F., 2002. Neural-network approximation of piecewise continuous functions: application to friction compensation. IEEE Trans. Neural Netw. 13 (May (3)), 745–751.
Sexton, R.S., Alidaee, B., Dorsey, R.E., Johnson, J.D., 1998a. Global optimization for artificial neural networks: a tabu search application. Eur. J. Oper. Res. 106 (April (2)), 570–584.
Sexton, R.S., Dorsey, R.E., Johnson, J.D., 1998b. Toward global optimization of neural networks: a comparison of the genetic algorithm and backpropagation. Decis. Support Syst. 22 (February (2)), 171–185.
Sexton, R.S., Dorsey, R.E., Johnson, J.D., 1999. Beyond back propagation: using simulated annealing for training neural networks. J. Organ. End User Comput. 11 (July (3)), 3–10.
Shang, Y., Wah, B., 1996. Global optimization for neural network training. Computer 29 (March (3)), 45–54.
Sharma, N., Arun, N., Ravi, V., 2013. An ant colony optimisation and nelder–mead simplex hybrid algorithm for training neural networks: an application to bankruptcy prediction in banks. Int. J. Inform. Decis. Sci. 5 (2), 188–203.
Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer. In: Proceedings of the IEEE International Conference Evolutionary Computation and IEEE World Congress Computational Intelligence, pp. 69–73.
Siddiqi, A.A., Lucas, S.M., 1998. A comparison of matrix rewriting versus direct encoding for evolving neural networks. In: Proceedings of the IEEE International Conference Evolutionary Computation and IEEE World Congress Computational Intelligence, pp. 392–397.
Sietsma, J., Dow, R.J., 1991. Creating artificial neural networks that generalize. Neural Netw. 4 (1), 67–79.
da Silva, A.J., Ludermir, T.B., de Oliveira, W.R., 2016. Quantum perceptron over a field and neural network architecture selection in a quantum computer. Neural Netw. 76, 55–64.
Silva, F.M., Almeida, L.B., 1990. Acceleration techniques for the backpropagation algorithm. In: Proceedings of the Neural Networks, ser. Lecture Notes in Computer Science, vol. 412. Springer, pp. 110–119.
Simovici, D.A., Djeraba, C., 2008. Mathematical Tools for Data Mining. Springer.
Sivagaminathan, R.K., Ramakrishnan, S., 2007. A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Syst. Appl. 33 (1), 49–60.
Slowik, A., 2011. Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training. IEEE Trans. Ind. Electron. 58 (August (8)), 3160–3167.
Socha, K., Blum, C., 2007. An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. Neural Comput. Appl. 16 (May (3)), 235–247.
Socha, K., Dorigo, M., 2008. Ant colony optimization for continuous domains. Eur. J. Oper. Res. 185 (March (3)), 1155–1173.
Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Inform. Process. Manag. 45 (July (4)), 427–437.
Sörensen, K., 2015. Metaheuristics–the metaphor exposed. Int. Trans. Oper. Res. 22 (1), 3–18.
Sporea, I., Grüning, A., 2013. Supervised learning in multilayer spiking neural networks. Neural Comput. 25 (2), 473–509.
Srinivas, M., Patnaik, L., 1991. Learning neural network weights using genetic algorithms-improving performance by search-space reduction. In: Proceedings of the International Jt. Conference Neural Networks, IJCNN, pp. 2331–2336.
Stanley, K.O., Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evol. Comput. 10 (June (2)), 99–127.
Steil, J.J., 2004. Backpropagation-decorrelation: online recurrent learning with O(N) complexity. In: Proceedings of the IEEE International Jt. Conference Neural Networks, vol. 2, pp. 843–848.
Stork, D.G., Walker, S., Burns, M., Jackson, B., 1990. Preadaptation in neural circuits. In: Proceedings of the International Jt. Conference Neural Networks, vol. 1, pp. 202–205.
Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. 11 (December (4)), 341–359.
Sum, J., Leung, C.-S., Young, G.H., Kan, W.-K., 1999. On the kalman filtering method in neural network training and pruning. IEEE Trans. Neural Netw. 10 (1), 161–166.
Tang, J., Deng, C., Huang, G.-B., 2016. Extreme learning machine for multilayer perceptron. IEEE Trans. Neural Netw. Learn. Syst. 27 (4), 809–821.
Tayefeh Mahmoudi, M., Taghiyareh, F., Forouzideh, N., Lucas, C., 2013. Evolving artificial neural network structure using grammar encoding and colonial competitive algorithm. Neural Comput. Appl. 22 (May (1)), 1–16.
Toh, K.-A., 2003. Deterministic global optimization for fnn training. IEEE Trans. Syst. Man Cybern. B Cybern. 33 (December (6)), 977–983.
Tong, D.L., Mintram, R., 2010. Genetic algorithm-neural network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection. Int. J. Mach. Learn. Cybern. 1 (September (1–4)), 75–87.
Trelea, I.C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf. Process. Lett. 85 (6), 317–325.
Trentin, E., Gori, M., 2001. A survey of hybrid ann/hmm models for automatic speech recognition. Neurocomputing 37 (1), 91–126.

Tsai, J.-T., Liu, T.-K., Chou, J.-H., 2004. Hybrid taguchi-genetic algorithm for global numerical optimization. IEEE Trans. Evol. Comput. 8 (August (4)), 365–377.
Tsai, J.-T., Chou, J.-H., Liu, T.-K., 2006. Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. IEEE Trans. Neural Netw. 17 (January (1)), 69–80.
Tsoulos, I., Gavrilis, D., Glavas, E., 2008. Neural network construction and training using grammatical evolution. Neurocomputing 72 (December (1)), 269–277.
Twomey, J., Smith, A., 1995. Performance measures, consistency, and power for artificial neural network models. Math. Comput. Model. 21 (1), 243–258.
Ulagammai, M., Venkatesh, P., Kannan, P., Padhy, N.P., 2007. Application of bacterial foraging technique trained artificial and wavelet neural networks in load forecasting. Neurocomputing 70 (October (16)), 2659–2667.
Van den Bergh, F., Engelbrecht, A.P., 2004. A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. 8 (3), 225–239.
Van den Bergh, F., Engelbrecht, A., 2001. Training product unit networks using cooperative particle swarm optimisers. In: Proceedings of the International Jt. Conference Neural Networks, vol. 1, pp. 126–131.
Vázquez, R.A., 2011. Training spiking neural models using cuckoo search algorithm. In: Proceedings of the IEEE Congress Evolutionary Computation (CEC), 2011, pp. 679–686.
Venkadesh, S., Hoogenboom, G., Potter, W., McClendon, R., 2013. A genetic algorithm to refine input data selection for air temperature prediction using artificial neural networks. Appl. Soft Comput. 13 (May (5)), 2253–2260.
Ventura, D., Martinez, T., 1998. An artificial neuron with quantum mechanical properties. In: Artificial Neural Nets and Genetic Algorithms. Springer, 482–485.
Vieira, S.M., Mendonça, L.F., Farinha, G.J., Sousa, J.M., 2013. Modified binary pso for feature selection using svm applied to mortality prediction of septic patients. Appl. Soft. Comput. 13 (8), 3494–3504.
van der Voet, H., 1994. Comparing the predictive accuracy of models using a simple randomization test. Chemom. Intell. Lab. Syst. 25 (2), 313–323.
Wand, Y., Wang, R.Y., 1996. Anchoring data quality dimensions in ontological foundations. Commun. ACM 39 (November (11)), 86–95.
Wang, L., Yang, B., Chen, Y., Zhao, X., Chang, J., Wang, H., 2012. Modeling early-age hydration kinetics of portland cement using flexible neural tree. Neural Comput. Appl. 21 (July (5)), 877–889.
Werbos, P.J., 1974. Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. dissertation, Harvard University.
Wessels, L.F., Barnard, E., 1992. Avoiding false local minima by proper initialization of connections. IEEE Trans. Neural Netw. 3 (November (6)), 899–905.
Weyland, D., 2010. A rigorous analysis of the harmony search algorithm: how the research community can be misled by a “novel” methodology. Int. J. Appl. Metaheuristic Comput. 1 (2), 50–60.
Whitley, D., Starkweather, T., Bogart, C., 1990. Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Comput. 14 (August (3)), 347–361.
Whitley, D., 1989. The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: Proceedings of the 3rd International Conference Genetic Algorithms, pp. 116–121.
Whitley, D., Hanson, T., 1989. Optimizing neural networks using faster, more accurate genetic search. In: Proceedings of the 3rd International Conference Genetic Algorithms, pp. 391–396.
Widrow, B., Lehr, M.A., 1990. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. Proc. IEEE 78 (9), 1415–1442.
Widrow, B., 1959. Adaptive sampled-data systems—a statistical theory of adaptation. In: IRE WESCON Convention Record, vol. 4, pp. 74–85.
Wiegand, S., Igel, C., Handmann, U., 2004. Evolutionary multi-objective optimisation of neural networks for face detection. Int. J. Comput. Intell. Appl. 4 (September (3)), 237–253.
Wilson, D.R., Martinez, T.R., 2003. The general inefficiency of batch training for gradient descent learning. Neural Netw. 16 (10), 1429–1451.
Wisrow, B., Hoff, M.E., et al., August 1960. Adaptive switching circuits. In: IRE WESCON Convention Record, vol. 4, pp. 96–104.
Wolpert, D.H., 1996. The lack of a priori distinctions between learning algorithms. Neural Comput. 8 (October (7)), 1341–1390.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (April (1)), 67–82.
Xi-Zhao, W., Qing-Yan, S., Qing, M., Jun-Hai, Z., 2013. Architecture selection for networks trained with extreme learning machine using localized generalization error model. Neurocomputing 102 (February), 3–9.
Yaghini, M., Khoshraftar, M.M., Fallahi, M., 2013. A hybrid algorithm for artificial neural network training. Eng. Appl. Artif. Intell. 26 (January (1)), 293–301.
Yang, J.-M., Kao, C.-Y., 2001. A robust evolutionary algorithm for training neural networks. Neural Comput. Appl. 10 (December (3)), 214–230.
Yang, X.-S., 2010. Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. 2 (2), 78–84.
Yang, X.-S., 2012. Flower pollination algorithm for global optimization. In: Proceedings of the Unconventional Computation and Natural Computation. Springer, pp. 240–249.
Yang, X.-S., Deb, S., 2009. Cuckoo search via lévy flights. In: Proceedings of the World Congress on Nature and Biologically Inspired Comput. NaBIC, pp. 210–214.
Yao, X., 1993. A review of evolutionary artificial neural networks. Int. J. Intell. Syst. 8 (4), 539–567.
Yao, X., 1999. Evolving artificial neural networks. Proc. IEEE 87 (September (9)), 1423–1447.
Yao, X., Liu, Y., 1997. A new evolutionary system for evolving artificial neural networks. IEEE Trans. Neural Netw. 8 (May (3)), 694–713.
Yao, X., Liu, Y., 1998a. Towards designing artificial neural networks by evolution. Appl. Math. Comput. 91 (1), 83–90.
Yao, X., Liu, Y., 1998b. Making use of population information in evolutionary artificial neural networks. IEEE Trans. Syst. Man Cybern. B Cybern. 28 (June (3)), 417–425.
Yao, X., Islam, M.M., 2008. Evolving artificial neural network ensembles. IEEE Comput. Intell. Mag. 3 (February (1)), 31–42.
Yao, Y., Rosasco, L., Caponnetto, A., 2007. On early stopping in gradient descent learning. Constr. Approx. 26 (April (2)), 289–315.
Yao, X., Liu, Y., 1996. Ensemble structure of evolutionary artificial neural networks. In: Proceedings of the IEEE International Conference Evolution Computational, pp. 659–664.
Ye, J., Qiao, J., ai Li, M., Ruan, X., 2007. A tabu based neural network learning algorithm. Neurocomputing 70 (January (4)), 875–882.
Yin, F., Mao, H., Hua, L., 2011. A hybrid of back propagation neural network and genetic algorithm for optimization of injection molding process parameters. Mater. Des. 32 (June (6)), 3457–3464.
Yusiong, J.P.T., Naval, P.C., Jr., 2006. Training neural networks using multiobjective particle swarm optimizationAdvances in Natural Computation. Springer, 879–888.
Zăvoianu, A.-C., Bramerdorfer, G., Lughofer, E., Silber, S., Amrhein, W., Klement, E.P., 2013. Hybridization of multi-objective evolutionary algorithms and artificial neural networks for optimizing the performance of electrical drives. Eng. Appl. Artif. Intell. 26 (8), 1781–1794.
Zhang, B.-T., Ohm, P., Mühlenbein, H., 1997. Evolutionary induction of sparse neural trees. Evol. Comput. 5 (June (2)), 213–236.
Zhang, G., 2000. Neural networks for classification: a survey. IEEE Trans. Syst. Man. Cybern. C Appl. Rev. 30 (November (4)), 451–462.
Zhang, J.-R., Zhang, J., Lok, T.-M., Lyu, M.R., 2007a. A hybrid particle swarm optimization and back-propagation algorithm for feedforward neural network training. Appl. Math. Comput. 185 (February (2)), 1026–1037.
Zhang, J.-R., Zhang, J., Lok, T.-M., Lyu, M.R., 2007b. A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. Appl. Math. Comput. 185 (2), 1026–1037.
Zhang, Y., Wu, L., Wang, S., 2010. Bacterial foraging optimization based neural network for short-term load forecasting. J. Comput. Inform. Syst. 6 (January (7)), 2099–2105.
Zhang, B.-T., Veenker, G., 1991. Neural networks that teach themselves through genetic discovery of novel examples. In: International Jt. Conference Neural Networks, IJCNN, pp. 690–695.
Zhao, Z., Zhang, Y., 2011. Design of ensemble neural network using entropy theory. Adv. Eng. Softw. 42 (October (10)), 838–845.
Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P.N., Zhang, Q., 2011. Multiobjective evolutionary algorithms: a survey of the state of the art: a survey of the state of the art. Swarm Evol. Comput. 1 (1), 32–49.
Zhou, Z.-H., Wu, J., Tang, W., 2002. Ensembling neural networks: many could be better than all. Artif. Intell. 137 (May (1)), 239–263.
Zikopoulos, P., Eaton, C., et al., 2011. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Osborne Media.