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

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)."