Parameter Tuning of MLP Neural Network Using Genetic Algorithms

Meng Joo Er and Fan Liu

Abstract. In this paper, a hybrid learning algorithm for a Multilayer Percept-rons (MLP) Neural Network using Genetic Algorithms (GA) is proposed. This hybrid learning algorithm has two steps: First, all the parameters (weights and biases) of the initial neural network are encoded to form a long chromosome and tuned by the GA. Second, as a result of the GA process, a quasi-Newton method called BFGS method is applied to train the neural network. Simulation studies on function approximation and nonlinear dynamic system identification are presented to illustrate the performance of the proposed learning algorithm.

Keywords: Genetic algorithms, Backpropagation, Function approximation, Nonlinear dynamic system identification.

1 Introduction

The most noticeable applications of Neural Networks (NNs) are in the areas of communication channel equalization, signal processing, pattern recognition, system identification, prediction and financial analysis, and process control [1~5]. Such a widespread use of NN is mainly due to its behavioral emulation of the human brain and its learning ability. A NN can be termed as a parallel and distributed process that consists of multiple processing elements. NNs have emerged as an important tool in terms of learning speed, accuracy, and noise immunity and generalization ability.

In this paper, the most popular models of NN, such as feedforward NNs are considered. Supervised Learning (SL) is one of the most effective weight training algorithms, whereby efforts are made to find an optimal set of connective weights for a NN according to some optimality criteria. One of the most popular SL training algorithms for feedforward NN is backpropagation (BP). The BP is a gradient descent search algorithm. It is based on minimization of the total mean square

Meng Joo Er · Fan Liu
School of Electrical and Electronic Engineering, Nanyang Technological Universigy
Singapore 639798, Singapore
{Emjer,liuf0009}@ntu.edu.sg

H. Wang et al. (Eds.): The Sixth ISNN 2009, AISC 56, pp. 121–130.
springerlink.com © Springer-Verlag Berlin Heidelberg 2009

error between the actual output and the desired output [6]. This error is used to guide the search of the BP algorithm in the weight space. The BP is widely used in many applications in that it is not necessary to determine the exact structure and parameters of NN in advance. However, the problem of the BP algorithm is that it is very often trapped in local minima and the learning and the adaptation speed is very slow in searching for global minimum of the search space. The speed and robustness of the BP algorithm are sensitive to several parameters of the algorithm and the best parameters vary from problems to problems.

Recently, some developed evolutionary algorithms, notably Genetic Algorithms (GA), have attracted a great attention in the NN areas [6]-[12]. The GA is proposed to train a feedforward NN and its performance is investigated in [6]. In [7], the GA is proposed to optimize the NN topology and connective weights. The GA is used to identify unimportant neurons and delete those neurons to yield a compact structure in [9]. By working with a population of solutions, the GA can seek many local minima, and thus increase the likelihood of finding global minimum. This advantage of the GA can be applied to NNs to optimize the topology and weight parameters.

In this paper, a hybrid algorithm based on GA technique is proposed to optimize parameters of the MLP Neural Network (MLPNN). A simple chromosome representation is used, which contains information about connections, weights and biases of the MLPNN. The parameter learning process, based on GA technique and BP algorithm, is a two-step learning process. In the first step, the initial parameters, such as weights and biases of the NN are tuned by the GA. In the second step, the BP algorithm and the quasi-Newton method is introduced to train the initial NN to yield optimal values of weights and biases of the NN.

This paper is organized as follows: In Section 2, the general structure of the MLPNN and the process of the GA optimal algorithm are briefly reviewed. Details of the proposed hybrid algorithm are given in Section 3. To demonstrate the effectiveness of the proposed algorithm, simulation studies on function approximation and nonlinear dynamic system identification are carried out in Section 4. Conclusions are given in Section 5.

2 MLPNN

MLPNN has become very popular in past decades. The development of the BP training algorithm represents a landmark in NNs in that it provides a computationally efficient method for training the MLPNN. A general three-layer MLPNN is depicted in Fig. 1.

In Fig. 1, the NN is a three-layer feedforward NN. Here, multi-input and single-output (MISO) systems are considered. However, all results could be extended to multi-input and multi-output (MIMO) systems. The mathematical description of the network is as follows.

Input layer: Each node represents an input variable p_i, $i = 1, 2, \cdots, n$.

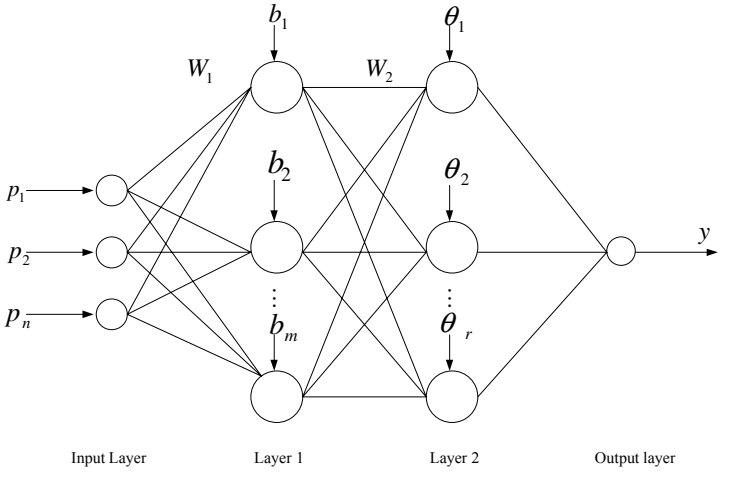

Fig. 1 Topology structure of a three-layer feedforward NN

Layer 1: Each node represents a neuron of layer 1. The term w_{ji} denotes the weight of the link between the *jth* neuron of layer 1 and the *ith* input variable. Here, W_1 is the corresponding weight matrix of layer 1.

Layer 2: Each node represents a neuron of layer 2. The term w_{kj} denotes the weight of the link between the *kth* neuron of layer 2 and the *jth* neuron of layer 1. W_2 is the corresponding weight matrix of layer 2.

Output layer: Each node represents an output as the summation of the incoming signals from layer 2.

In such a network, a neuron sums a number of weighted inputs and a bias and then it passes the result through a nonlinear activation function. Fig. 2 shows a typical structure of a MLP neuron with n -input connections and a single output.

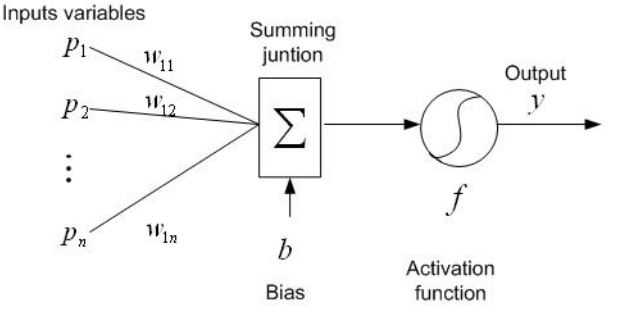

Fig. 2 A typical MLP neuron

The output of the neuron is given by

$$y = f(\sum_{i=1}^{n} w_{1i} p_i + b), \tag{1}$$

where

$p_1, p_2, \cdots, p_n$: Input variables;
$w_{11}, w_{12}, \cdots, w_{1n}$: Connective weights;
b : Bias;
f : Activation function.

Before a NN can be used for any purpose, the weights of neurons and bias values should be adjusted by a learning algorithm so that outputs of the NN will match the desired patterns for specific sets of inputs. One of the most widespread learning algorithms is the BP algorithm. However, difficulties faced by the BP algorithm are poor convergence rate [13] and an entrapment of local minimum. Here, we choose a quasi-Newton method called the BFGS method for NN training [14].

3 Hybrid BP Learning Algorithm Based on GA

The GA is a derivative-free stochastic optimization method based on the features of natural selection and biological evolution. It has several advantages over other optimization algorithms. It can be applied to both continuous and discrete optimization problems. Compared with the BP algorithm, the GA is less likely to get trapped in local minima [6]. It is a computational model inspired by population genetics. It has been used mainly as function optimizers and it has been demonstrated to be an effective global optimization tool, especially for multi-model and non-continuous functions.

The GA evolves a multi-set of elements, called a population of individuals. Each individual $X_i (i = 1,2,\cdots, p)$ (p, the size of the population) of population X represents a solution of the problem. Individuals are usually represented by strings and each element of which is called a gene. The value of a gene is called its allelic value, and its range is usually restricted to [0, 1], but it can also be continuous and even structured. We use real-valued strings in our approach. The GA is capable of maximizing a given fitness function F computed on each individual of the population.

The block diagram of the proposed hybrid algorithm is depicted by Fig. 3. This model describes a hybrid learning algorithm of MLPNN by using the GA to optimize the parameters of the network. All the parameters of the network are encoded to form a long chromosome and tuned by the GA. Then, as a result of the GA process, the BP algorithm is used to train the network. The procedure of the hybrid BP learning algorithm is presented as follows.

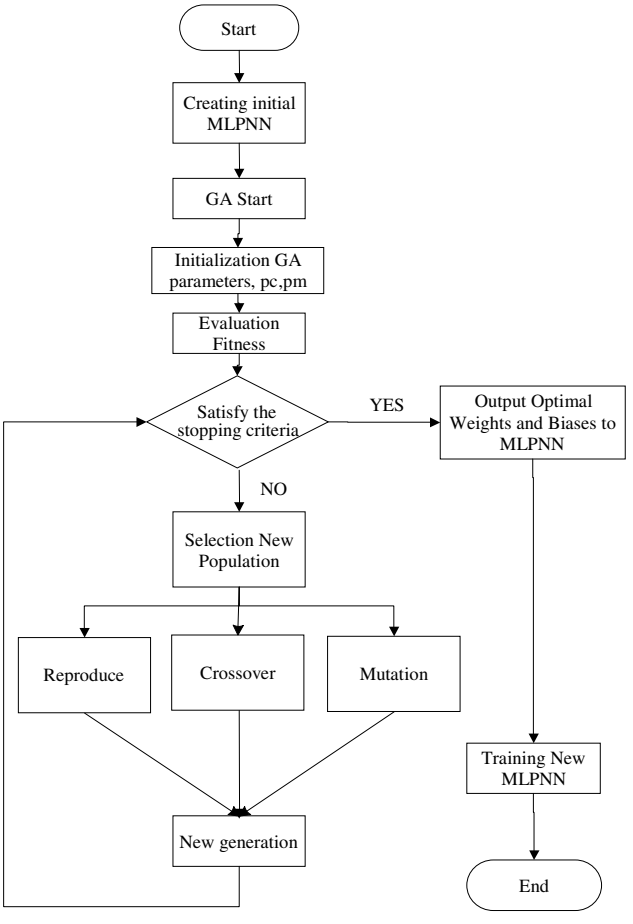

Fig. 3 Flowchart of the proposed learning algorithm method

3.1 Chromosome Representation

A MLPNN can be represented by a directed graph, encode on a chromosome with each parameter (weights and biases). All these parameters are memorized by a row vector $C = (c_i), i = 1,2,\cdots,N$, where N is the number of all NN parameters. We can write the chromosome as

$$C = [W_1, W_2, b_1, b_2, \cdots, b_m, \theta_1, \theta_2, \cdots, \theta_r], \tag{2}$$

where W_1 denotes the connective weight of link between the input layer and the first hidden layer, W_2 is the connective weight of link between the first hidden layer and the second hidden layer, $b_1, b_2, \cdots, b_m$ are the biases of neurons of the

first hidden layer, $\theta_1, \theta_2, \cdots, \theta_r$ are the biases of neurons of the second hidden layer. We use the real-value encoding in this paper, W_1 , W_2 , $b_1, b_2, \cdots, b_m$, $\theta_1, \theta_2, \cdots, \theta_r$ are the real values of the connective weights, biases respectively.

3.2 Fitness Function

The fitness function is dependent on problem and is used to evaluate the performance of each individual. The error signal of the output neuron j at iteration n (i.e., presentation of the nth training example) is defined by

$$e_j(n) = d_j(n) - y_j(n) . \tag{3}$$

We defined the instantaneous value of the error energy for neuron j as $\frac{1}{2}e_j^2(n)$. Correspondingly, the value of $\xi(n)$ is obtained by summing $\frac{1}{2}e_j^2(n)$ over all neurons in the output layer

$$\xi(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) , \tag{4}$$

where the set C includes all the neurons in the output layer of the network. For MLPNN it is the sum squared error.

The fitness is defined as by summing $\xi(n)$ over all n with respect to the set size N , as shown by

$$F = \sum_{n=1}^{N} \xi(n) . \tag{5}$$

Obviously the objective is to minimize $F(\cdot)$ subject to weights w_{ji}, w_{kj} and biases $b_i (i = 1,2,\cdots,m)$ and $\theta_j (j = 1,2,\cdots,r)$.

3.3 Selection

Selection operator is to select individuals from the population for reproduction based on the relative fitness value of each individual. The extraction can be carried out in several ways. We use "NormGeomSelcet" ranking selection method in this paper, it is a ranking selection function based on the normalized geometric distribution.

3.4 Crossover

To apply the standard crossover operator the individuals of the population are randomly paired. The two mating chromosomes are cut once at corresponding points and the sections after the cuts exchanges. The crossover point can be chosen randomly.

3.5 Mutation

After crossover, the new individuals are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum. A variable is selected with a certain probability and its value is modified by a random value. Here, we choose non-uniform mutation method. Non-uniform mutation changes one of the genes of the parent based on a non-uniform probability distribution.

4 Illustrative Examples

One of the most powerful uses of an NN is function approximation. Two examples are simulated in this section to demonstrate the effectiveness of the proposed hybrid learning algorithm. They are the Hermite polynomial function and the nonlinear dynamic system identification.

4.1 Function Approximation

Here, the underlying function to be approximation is the Hermite polynomial, which is defined as:

$$f(x) = 1.1(1 - x + 2x^2)\exp(-\frac{x^2}{2}) . \tag{6}$$

A set of 100 one-input normalized data and the corresponding target data is used as the training data.

Another 100 input-target data are also used as the testing data. The parameter of GA is set as, crossover probability: $p_c = 0.09$, number of generation: $gen = 100$. We construct a 1-15-1 MLPNN and the activation functions are set as "tansig", "tansig" and "purelin" respectively. "Tansig" denotes hyperbolic tangent sigmoid transfer function. "Purelin" denotes linear transfer function. A set of 17 neurons is generated. The total number of parameters is 46. Simulation results are demonstrated in Fig. 4 ~ Fig. 6. The comparisons of the proposed method and BP method are depicted in Fig. 4 and Fig. 5. The mean squared error arrived at goal value at the 121 epoch in Fig. 4 and at 123 epoch in Fig. 5. The fitness value is shown in Fig. 6.

4.2 Nonlinear Dynamic System Identification

The system can be described as

$$\begin{aligned} y(t+1) &= \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} + u(t) \\ y(1) &= 0, u(t) = \sin(2\pi t/25) \end{aligned} \tag{7}$$

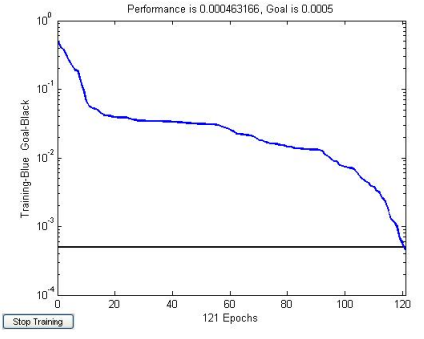

Fig. 4 Training performance with GA

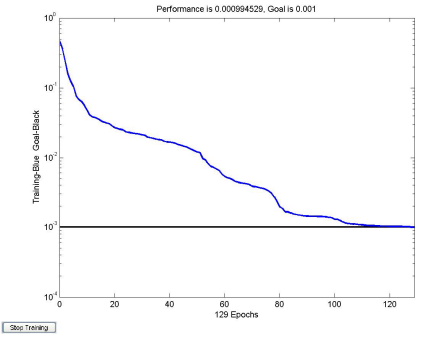

Fig. 5 Training performance without GA

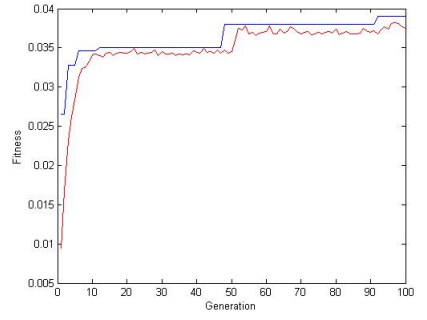

Fig. 6 Fitness evolution (Example 1)

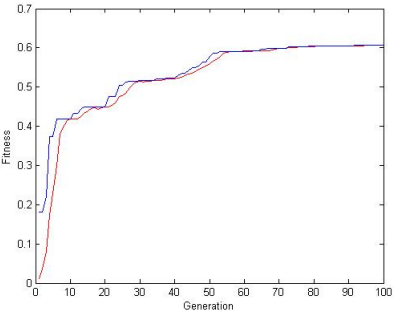

Fig. 7 Fitness evolution (Example 2)

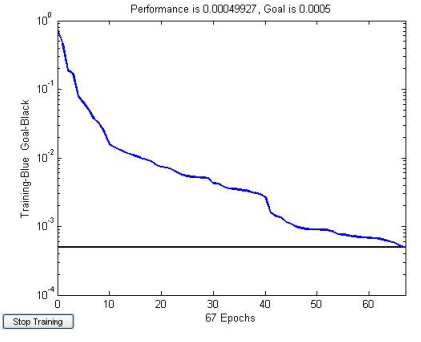

Fig. 8 Training performance with GA

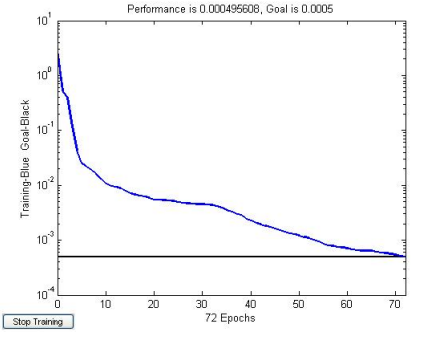

Fig. 9 Training performance without GA

The model is identified in series-parallel mode and is given by

$$\hat{y}(t+1) = f(f(t), y(t-1), u(t)) . \tag{8}$$

It is a three-input-single-output model. There are 200 input-data sets chosen as training data. The parameter of GA is set as, crossover probability: $p_c = 0.095$, number of generation: $gen = 100$. We construct a 3-3-1 MLPNN and the activation functions are set as "tansig", "tansig" and "purelin" respectively. A set of 7 neurons is generated. The total number of parameters is 16. Simulation results are demonstrated in Fig. 7 ~ Fig. 9. The comparisons of the proposed method and BP method are depicted in Fig. 8 and Fig 9. The mean squared error arrived at goal value at the 67 epoch in Fig. 7 and at about 71 epoch in Fig. 9. The fitness value is shown in Fig. 7.

5 Conclusions

In this paper, a hybrid learning algorithm using GA to optimize parameters of MLPNN is presented. The initial structure of the MLPNN is created, and all the parameters of the network are tuned by GA. Simulations show that the hybrid learning algorithm has good performance in function approximation and nonlinear dynamic system identification. The GA, as a global search tool, can avoid the local minima problem of the BP. In our recent works, we propose that GA can be used to optimize the parameters of MLPNN. Further research will focus on how to use the GA to obtain a more compact structure of the network with higher accuracy. Not only the parameters of the network, but also the network topology can be optimize by the GA is a future research work.

References

1. Kadirkamanathan, V., Niranjan, M.: A Function Estimation Approach to Sequential Learning with Neural Networks. Neural Computation 5, 954–975 (1993)
2. Juang, C.F., Chin, C.T.: An On-Line Self-Constructing Neural Fuzzy Inference Network and its Applications. IEEE Trans. Fuzzy Systems 6, 12–32 (1998)
3. Widrow, B., Rumelhart, D.E., Lehr, M.A.: Neural Networks Applications in Industry, Business and Science. Communication of the ACM 37, 93–105 (1994)
4. Narendra, K.S., Kannan, P.: Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. Neural Networks 1, 4–27 (1990)
5. Vellido, A., Lisboa, P.J.G., Vaughan, J.: Neural Networks in Business: A survey of applications. Expert Systems with Applications 17, 51–70 (1999)
6. Siddique, M.N.H., Tokhi, M.O.: Training Neural Networks: Backpropagation vs Genetic Algorithms. In: Proceeding of the International Joint Conference on Neural Networks, vol. 4, pp. 2673–2678 (1999)
7. Tang, K.S., Chan, C.Y., Man, K.F., Kwong, S.: Genetic Structure for NN Topology and Weights Optimization. In: IEEE Conference Publication, vol. 414, pp. 250–255 (1995)
8. Leng, G., McGinnity, T.M., Prasad, G.: Design for Self-Organizing Fuzzy Neural Networks Based on Genetic Algorithms. IEEE Trans. on Fuzzy Systems 14, 755–765 (2006)

9. Chen, S., Wu, Y., Luk, B.L.: Combined Genetic Algorithm Optimization and Regularized Orthogonal Least Squares Learning for Radial Basis Function Networks. IEEE Trans. on Neural Networks 10, 1239–1243 (1999)
10. Maniezzo, V.: Genetic Evolution of Topology and Weight Distribution of Neural Networks. IEEE Trans. on Neural Networks 5, 39–53 (1994)
11. Billings, S.A., Zheng, G.L.: Radial Basis Function Network Configuration Using Genetic Algorithms. Neural Networks 8, 877–890 (1995)
12. Frank, H., Leung, F., Lam, H.K., Ling, S.H., Peter, K., Tam, S.: Tuning of the Structure and Parameters f a Neural Network Using an Improved Genetic Algorithm. IEEE Trans. on Neural Networks 14, 79–88 (2003)
13. Lippmann, R.P.: An Introduction to Computing with Neural Nets. IEEE ASSP Magazine 1, 4–22 (1987)
14. Watrous, R.L.: Learning Algorithms for Connectionist Networks: Applied Gradient Meth-ods for Nonlinear Optimization. In: Proceeding IEEE First Int. Conf. Neural Net., vol. 2, pp. 619–627 (1987)