OPTIMIZED BACK-PROPAGATION ALGORITHM FOR PATTERN RECOGNITION

Musbah J. AQEL [1], Ziad AL-QADI[2]
[1]Department of Electrical and Computer Engineering
Faculty of Engineering, Applied Science University, Amman, Jordan
[2]Department of Computer Engineering,
College of Engineering, Al-Balqaa Applied University, Amman, Jordan
Email: musbahaqel@yahoo.com

Keywords: Back propagation, Genetic algorithms, artificial neural network

Abstract. *In this paper, a genetic algorithm (GA) has been proposed as an optimization method in order to evolve the artificial neural network with the aim of optimizing connection weights and network architecture for a pattern recognition and a classification tools. The proposed algorithm has been applied and tested for optimizing connection weights in recognition of a different group of patterns. Also, it has been studied to implement the optimization of the network architecture. Several training back-propagation algorithms are tested in an attempt to find an ideal artificial neural network (ANN) training algorithm in pattern recognition and classification. Tests have shown that the training optimized by GA method is faster than the BP training algorithm and produces a lower SSE in a smaller number of epochs.*

1. INTRODUCTION

Artificial neurons are computational models based on the biological neuron. Artificial neural networks (ANNs) are interconnected networks of artificial neurons. By the use of training, generally through a gradient descent algorithm such as Back-propagation (BP) which is considered the most effective method for the training of the multilayer perceptron MLP [3,4,5]. ANNs are able to learn and generalize input patterns to output patterns [2].

The ability of an ANN to learn is considered to be a property of its structure as well as the value of its weights, however the most appropriate ANN structure is still generally heuristically chosen for an application.

The genetic algorithm is one of a relatively new class of stochastic search algorithms. Stochastic algorithms are those that use probability to help guide their search [6]. Genetic algorithms require the use of special operators in order to simulate the evolutionary processes which they emulate [7]. After many generations, the GA will produce an optimized or best possible solution [8].

Several approaches have been applied to combining genetic algorithms and neural networks. The most successful of these are hybrid algorithms that combine a genetic algorithm to search for optimal ANN architectures with a gradient-descent based training algorithm to optimize the ANN weights. Their success lies in the combination of efficient global search with efficient local search.

Gradient-descent ANN training algorithms require the error of an ANN over the input training patterns to be computable. The error of an ANN is the result of a mean or square of the sum of the difference between the actual and the expected outputs. It requires the expected output to be known, which makes it most suitable to supervised learning tasks.

In ANN/genetic algorithm approaches applied to a supervised learning, this error value is often used as the fitness function. However, this is not a mandatory requirement for genetic algorithms, where the fitness function can be any appropriate measure of fitness. This means that

successful algorithms that can evolve neural networks that are not dependent on gradient information have a wide applicability to domains other than supervised learning. This is especially important to reinforcement learning tasks such as robot controllers [13].

The fitness function that has been used in this paper is a traditional ANN error calculation, where the error of each training pattern is squared, and then the sum of these values over a number of training patterns is calculated.

This paper aims to develop an algorithm using a genetic approach, to simultaneously evolve feed forward artificial neural network architectures and weights with or without incorporating a gradient-descent learning process such as back propagation. The performance of the evolved neural networks will be tested by applying the algorithm to a range of patterns classification and recognition problems. The problems that have been chosen for this are English capital letters data classification.

2. THE PROPOSED ALGORITHM

The proposed algorithm is developed using a genetic approach, to simultaneously evolve feed forward artificial neural network architectures and weights with or without incorporating a gradient –descent process such as back propagation.

The genetic algorithm consists of the evaluation of individuals, selection of individuals which will contribute to the next generation, recombination of the parents by means of crossover, mutation and other operators to produce a new generation. In this process, selection has the role of guiding the population towards some optimal solution, crossover the role of producing new combinations of partial solutions, and mutation the production of novel partial solutions. Based on these basic principles of genetic algorithms, the proposed algorithm proceeds as follows:

Place t = 0;
1. Generate Initial Population (Pt)
2. Evaluate (Pt)
3. Test the stopping conditions whether they are met or not
If no go to step (4) otherwise go to step (5)
4. Do the following and then
I. Select chromosome method and reproduction
II. Crossover operation
III. Mutation operation
IV. Go to step (2).
5. Apply forward pass of the ANN BP using the optimum chromosome.
6. Test if SSE is acceptable
If yes then stop, otherwise go to step (7)
7. Train the ANN by BP algorithm using the optimum chromosome as a starting point (i. e. initial weight values) then go step (6)

3. PROPOSED ALGORITHM IMPLEMENTATION

In this work, the proposed genetic algorithm shown in figure (1) has been used as an optimization method in order to evolve the artificial neural network with the aim of optimizing connection weights and network architecture for a pattern recognition and a classification tool.

The proposed algorithm has been applied and tested for optimizing connection weights in a recognition of a different group of patterns such as the classification and recognition between the letters (A, B, C), as well as the classification and recognition between the letters (D, E, F), which can be extended to another groups of patterns. Also the proposed algorithm has been developed to implement the optimization of the network architecture.

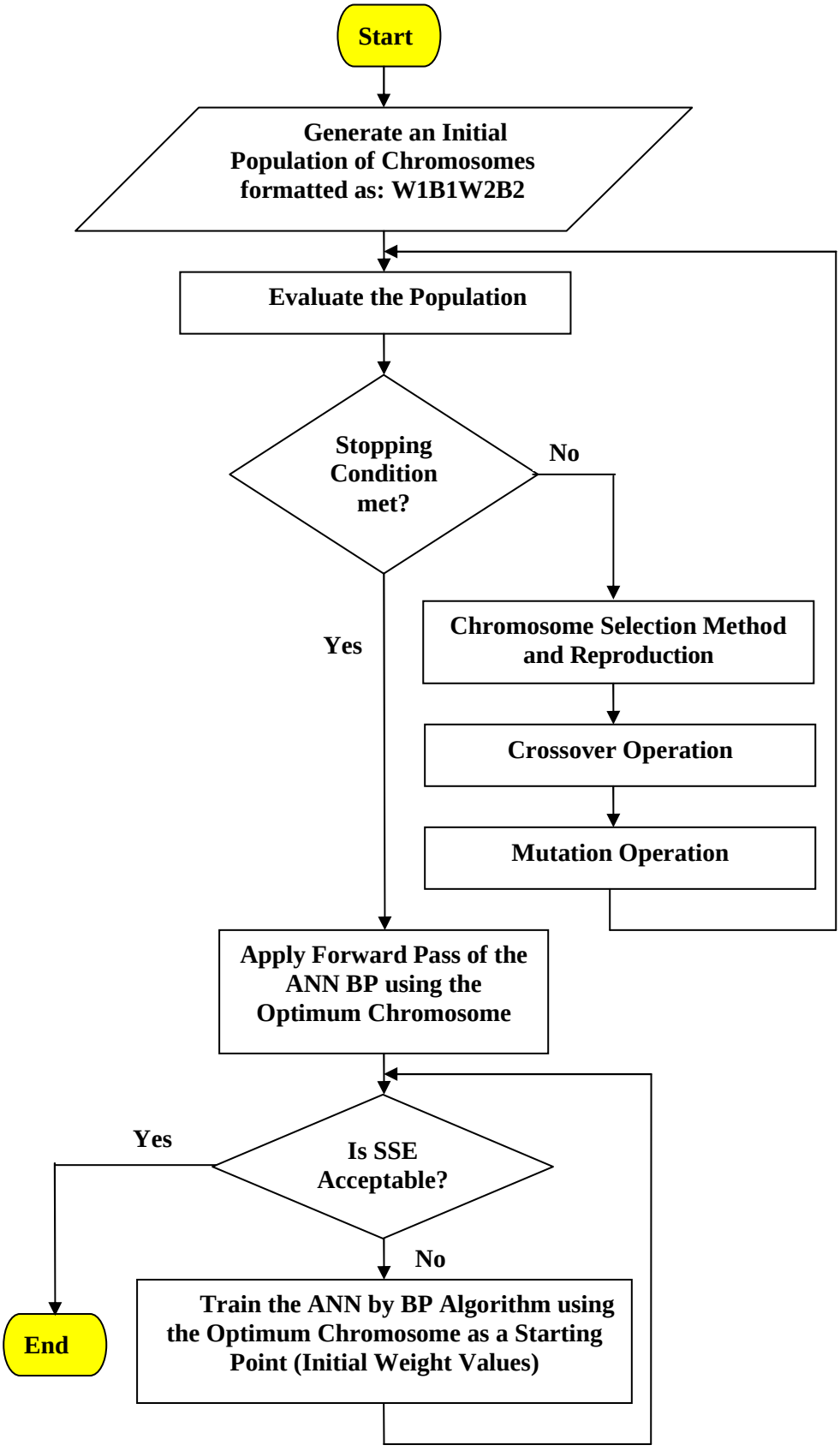

Fig. 1 The proposed algorithm

This algorithm has been implemented for a multilayered artificial neural network to be an optimization method as shown below: The implementation process can be described as follows:

1. Generate Initial Population (Pt)

A group of chromosomes can be regarded to present the initial population. So, the population can be defined as an array of M chromosomes, initially these chromosomes are randomly created.

In this work, first, a network of three layers with BP has been tried. The first layer is input layer and the number of nodes in this layer depends on the size of input.

The second layer is the hidden layer and it contains the hidden nodes which can be chosen in the training operation; while the last one is the output layer and the number of nodes at this layer depends on the number of patters which will be learned.

The format of chromosomes

Each chromosome is a vector of one dimension contains N elements. These elements represent the weight connections and the bias connections in the neural network. Length of this vector is changed depending on the number of input, hidden and output neurons. So to compute the length of this vector, the following equation is used:

$$N = (I * H) + H + (H * O) + O$$

Where:

N: is the length of the chromosome which represents the number of elements.

I: is the number of input nodes.

H: is the number of hidden nodes.

O: is the number of output nodes.

As an example, for three output patterns, three output nodes required; for each pattern is an array of two dimensions with size (5*7), the numbers of input nodes are 35 nodes, and if the hidden layer contains 3 nodes, then the number of elements of each part in the chromosome are:

$W_1 = 35*3 = 105$

$B_1 = 3$

$W_2 = 3*3 = 9$

$B_2 = 3$

The length of the chromosome is: $W_1+B_1+W_2+B_2 = 105+3+9+3 = 120$.

All the connection weights have represented in this chromosome have real number values.

2. Evaluate the Population (Pt)

For each chromosome in the initial population, fitness value has been computed by applying the feed-forward pass of the back propagation algorithm.

The most suitable fitness measure for optimization ANN is the Sum Square Error (SSE) and Mean Square Error (MSE) for each chromosome.

The network (weights) that produces minimum SSE or minimum MSE is the chromosome that produces maximum fitness according to the fitness equation used in the algorithm such as:

Fitness+ 1/*error*

Or:

Fitness = 1/ (1+ *error*)

Where: *Error* is SSE or MSE.

3. Implement the Genetic Operators

There is more than one method used to check convergence or stopping condition, one of them is to check the score of the best individual over a number of generations. Also, if the percentage of individuals in the population are identical (or similar) to each other, as well as if the maximum number of generation reached.

3.1. Chromosome selection method and Reproduction

It is very important to use a suitable selection method for next steps. This method depends on the fitness values of the chromosomes. Generally, the chromosomes with higher fitness will be repeated in more than one place at the selection set. For selection we create a "mating pool" which is the set of chromosomes that are selected for mating, with probability Pi where:

$$P_i = f_i / \sum f_i$$

3.2. Crossover operation

Crossover operation allows new individuals to be created. The operation starts with two parents and produces two offspring. The two offspring are usually different from their parents and different from each other. The crossover operation is controlled by a crossover probability. The crossover probability value is generally large for example, let the crossover probability value (0.7).

As shown in the structure of the neural network, there are connection weights from all nodes in the input layer to each node in the hidden layer, and also there are connection weights from all hidden nodes in the hidden layer

to each nodes in the output layer, so the crossover operation are applied under the control of crossover probability value.

For each hidden nodes (output nodes) in both chromosomes (pair of chromosomes) the following steps are repeated:

-A random number should be created, if this number will satisfy the crossover probability, then the crossover operation will be continued.

- All connections and their values of the hidden nodes of the hidden layer should be swapped (output nodes at the output layer) in both chromosomes to generate a new set of chromosomes with new connection weight values.

4. Mutation operation

In the mutation, a single chromosome is chosen, and set a random real value for a certain hidden (output) weight under the control of the mutation probability.

The frequency of applying the mutation operation is controlled by a parameter called mutation probability. The mutation probability is a small value, for example, it may equal to 0.2.

After completing these steps a new population will be generated, compute the fitness for all new chromosomes and return to step 3.1.

The genetic algorithm will produce the best chromosome which has a maximum fitness or according to neural network measure it has a minimum output error compared with a desired output.

Since this chromosome represents the network weights, if its output error was acceptable value; then it can be used as a solution map for the input/output training pairs of the application.

If its output error was not acceptable value, and this chromosome is not enough to be as a classifier net; then it can be used as a starting point with an initial weights values for the training algorithm of the multilayer neural network.

4.1. Genetic Optimization for Network Structures

As in most neural network applications, the architecture of the network chosen can determine the success or failure of the application, the most important parameter in this architecture is the number of hidden layer and the number of nodes at each hidden layer.

So, here the genetic algorithm is used to find the optimal structure of an Artificial Neural Network where its structure can be described as follow:

The first layer is an input layer and the number of nodes at this layer depends on the length of patterns to be learned. As shown in figure (3.1) the pattern is an array of two dimensions (5*7), the number of nodes in the input layer are 35 nodes. The second layer may be one or more hidden layers, while the last one is the output layer and the number of node at this layer depends on the number of patterns to be trained (classified).

The same proposed algorithm for optimizing network weights can be used with another chromosome format according to the optimizing of the network structure as follow:

Each chromosome is a vector of one dimension contains three elements for three hidden layer, each element contains the number of hidden nodes in that layer.

4. EXPERIMENTAL RESULTS

The network is connectionless (no connection establishment and no resource reservation). The data is transferred as discrete packets. The network nodes forward packets as well as possible. They may be lost, delayed or their order may change.

There are two tests that are carried out using the proposed algorithm:

1. Test (1): patterns recognition of the letters (A,B,C):
2. Test (2) :Patterns recognition of the letters (D,E,F):

4.1. Test (1): patterns recognition of the letters (A, B, C)

In this test, a neural network of (35) input neurons, (3) hidden neurons and (3) output neurons was trained on sets of references patterns of the letters (A, B, C).

Both algorithms were allowed to run until either the SSE was less than or equal to predetermined value ($10^{\wedge -5}$) or the maximum number of training iterations had occurred.

In this test a version of genetic optimization method was used that replaces the least fit weight chromosome by using the most important genetic operations such as crossover operation with a large value of probability (0.7) and mutation operation with a small value of

probability (0.2), also the elitism operation was performed to improve the GA's performance.

During the training by using back-propagation algorithm there are some problems have been noticed, on the contrary to what results from using BP algorithm in training, such as divergence and local minima as shown below:

a) At training the network of (35-19-3) with a purelin-purelin activation functions in the hidden-output neurons respectively. It has been noticed that the network failure due to converge.

As the network gets trained, the total input of a hidden unit or output unit can therefore reach very high (either positive or negative) values; the weights can be adjusted to very large values. This will cause the network failure to convergence. This problem can be solved by using the sigmoid activation function; because of the units with a sigmoid activation function will have activation very close to zero or very close to one.

b) The instability in the SSE curve which called local minima. This is considered as a one of the great disadvantages of the BP which is eliminated by using GA.

The effect of several parameters on the performance of the network, when it is trained by using back-propagation algorithm, will be investigated later, also there will be a demonstration of how using genetic algorithm in optimizing the network led to improve it's performance and reducing the disadvantages resulting from the influence of these parameters and eliminating the influence of some parameters.

From these parameters, there was a study of the effect of the number of hidden neurons as shown below.

The effect of the number of hidden neurons:

In this section, the effect of selection the number of hidden neurons on the performance of the network has been investigated. If it is selected too small, the function to be learnt may not possibly be represented, as the capacity of the network is not sufficient. If the number of hidden neurons is increasing, the computing time of the learning phase also increased.

By using GA, this effect can be reduced according to the property of a global search that aides the increasing of hidden neurons; as shown in table 1.

Table 1: SSE training for different hidden neurons

Number of hidden neuron	Training by BP		Training with GA aid	
	Number of Iterations	SSE	Number of Iterations	SSE
19	248	10^{-5}	162	10^{-5}
6	94	10^{-5}	35	10^{-5}
3	120	10^{-5}	70	10^{-5}

Comparison of training back-propagation algorithms:

This work also compared with several training back-propagation algorithms in an attempt to find an ideal artificial neural network (ANN) training algorithm in pattern recognition and classification using the MATLAB Neural Network Toolbox.

Several neural networks were created and trained using the same number of neuron, layers, epochs, performance goal and various training algorithms. A 35-3-3 network, with logsig, purline functions was used. These parameters were chosen by trial and error as is usually done in the training of a neural network.

There are several variations to the basic training algorithm of the back propagation neural network provided by MATLAB that has been used in this research.

- TRAINBP: Train feed-forward network with back-propagation (BP).
- TRAINBPA: Train feed-forward network with BP and adaptive learning rate.
- TRAINBPM: Train feed-forward networks with BP and momentum term.
- TRAINBPX: Train feed-forward network with fast back-propagation with adaptive learning rate and momentum term. Table (2) summarizes the initial results obtained from the training of the network by BP algorithm for the input training patterns.

Table2. Summary of the initial results obtained by BP

Algorithm	Iterations	Training Error
trainbp	228	$9.39*10^{-6}$
trainbpa	300	No training
trainbpx	48	$9.47*10^{-6}$
Trainbpm	103	$4.2*10^{-6}$

Table 3 summarizes the same training with aided of genetic algorithm that used to optimizes the network weights as an optimum starting point for training with BP algorithm.

Table3: Summary of the initial results obtained by BP aided with GA

Algorithm	Iterations	**Training Error**
GA+trainbp	44	$9.57*10^{-6}$
GA+trainbpx	24	$5.91*10^{-6}$
GA+trainbpm	70	$7.34*10^{-6}$

4.2. Test (2): Patterns recognition of the letters (D, E, F)

In this test a neural network of (35) input neurons, (3) hidden neurons and (3) output neurons was trained on sets of references patterns of the letters (D, E, F) as shown in figure 2.

Both algorithms were allowed to run until either the SSE was less than or equal to predetermined value (10^-5) or the maximum number of training iterations had occurred.

In this test, a version of genetic optimization method was used that replaces the least fit weight chromosome by using the most important genetic operations such as crossover operation with a large value of probability (0.7) and mutation operation with a small value of probability (0.2), also the elitism operation was performed to improve the GA's performance.

Table 4 shows the sum square error values with number of epochs, as a result for the training of the network (35-3-3) with the reference training patterns of letters (D, E, F), these results obtained for a ten different training times by using a classical back-propagation algorithm, and the same training with aided of genetic algorithm that used to optimizes the network weights as an optimum starting point for training with BP algorithm.

```
[-1 -1 -1 -1  1;        [-1 -1 -1 -1 -1;        [-1 -1 -1 -1 -1;
 -1  1  1  1 -1;         -1  1  1  1  1;         -1  1  1  1  1;
 -1  1  1  1 -1;         -1  1  1  1  1;         -1  1  1  1  1;
 -1  1  1  1 -1;         -1 -1 -1 -1  1;         -1 -1 -1  1  1;
 -1  1  1  1 -1;         -1  1  1  1  1;         -1  1  1  1  1;
 -1  1  1  1 -1;         -1  1  1  1  1;         -1  1  1  1  1;
 -1 -1 -1 -1  1];        -1 -1 -1 -1 -1];        -1  1  1  1  1];
```

Fig. 2a. The reference patterns for the letters D, E, F

TRAINING INPUT **TARGET OUTPUT**

```
-1    -1    -1
-1    -1    -1
-1    -1    -1
-1    -1    -1
-1    -1     1
-1    -1    -1
 1     1     1
 1     1     1
 1     1     1      [-1  -1   1 ;
 1     1    -1
-1    -1    -1       -1   1  -1 ;
 1     1     1
 1     1     1        1  -1  -1 ];
 1     1     1
 1     1    -1
-1    -1    -1
-1    -1     1
-1    -1     1
 1    -1     1
 1     1    -1
-1    -1    -1
 1     1     1
 1     1     1
 1     1     1
 1     1    -1
-1    -1    -1
 1     1     1
 1     1     1
 1     1     1
 1     1    -1
-1    -1    -1
 1    -1    -1
 1    -1    -1
 1    -1    -1
 1    -1     1
```

Fig. 2b. The training data and output for letters D, E, F

Training with BP		Training by BP optimized by GA	
No. of Iterations	SSE*10^-6	No. of Iterations	SSE*10^-6
111	9.04	53	9.06
101	9.20	48	7.95
96	9.40	44	9.66
213	9.81	46	9.63
207	9.94	51	9.87
195	9.63	54	8.30
214	9.64	75	9.10
106	8.46	84	8.40
191	9.78	76	9.71
243	9.88	68	7.74

Table 4 Comparison between BP training and BP with GA optimization

From table 4, it's cleared that the training aided by GA is significantly faster than the BP method and produces a lower SSE in a smaller number of training epochs.

5. RESULT ANALYSIS

Tests show that the training optimized by GA method is faster than the BP training algorithm and produces a lower SSE in a smaller number of epochs.

In GAs there are different types of random operations which can introduce good solutions to the desired problem in a random method so GAs are free from Local Minima.

In GA the connection weight are created randomly and the GA operations depends on the natural selection (crossover, mutation), so there is no specific steps used to train ANN to strength the connection weights, as in classical algorithm, which has a specific steps to strength the connection weights. So the time required to train an ANN to recognize a specific number of input patterns using GA are unpredictable and it can take different time periods for different GA runs (for the same number of patterns).

Sometimes it could happen that though the ANN could theoretically solve a certain classification problem when it trained using GA, but it will not return a correct solution at all the time. This is because of the random nature of the algorithm and its reliance on natural selection, mutation, and crossover.

GA may not effectively be able to train multilayer neural networks, whose chromosome representation of its weights is large, because of the time required for training optimizes by GA is may be larger than the time required to learn the same patterns using classical BP algorithm.

6. CONCLUSION

A genetic algorithm was proposed as an optimizing method to evolve the artificial neural network with the aim of optimizing connection weights and network architecture for a pattern and a classification tools. The genetic algorithm produced the best chromosome which has a maximum fitness or according to neural network measure which it has a minimum output error compared with a desired output. Since this chromosome represents the network weights, if its output error was acceptable value, then it can be used as a solution map for the input/output training pairs of the application. The effect of several parameters (e.g., the number of hidden neurons) on the performance of the network, when it is trained by using back-propagation (BP) algorithm, have been investigated, also there was a demonstration of how using genetic algorithm in optimizing the network which led to improve it's performance and reducing the disadvantages resulting from the influence of these parameters.

References

[1].T. Binos," Evolving Neural Network Architecture and Weights Using an Evolutionary Algorithm ", M.Sc. thesis, Department of Computer Science, RMIT University, 2003.

[2]. B Krose, and P. Van Der Smagt, "An Introduction to Neural Networks", University of Amsterdam, 1996.

[3]. W. Kinnebrock, "Neural Networks-Fundamental, Applications, Examples", Oldenburg, 1995.

[4]. W. Y. Jen, and C. T. Lin, "A Second Order Learning Algorithm for Multilayer Networks Based on Block Hessian Matrix", Neural Network, vol. 11, No. 9, pp. 1607-1622, 1998.

[5]. S. Y. Lee, and S. H. Oh, "A New Error Function at Hidden Layers for Fast Training of Multilayer Perceptrons", IEEE Transactions on Neural Networks, vol. 10, No. 4, pp. 960-964, 1999.

[6]. K. Grant, "An Introduction to Genetic Algorithms", C/C++ users Journal, pp. 45-58, March 1995.

[7]. A. D. Sofge, L. D. Elliott, "Improved Neural Modeling of Real World Systems using Genetic Algorithm Based Variable Selection", International Conference on Neural Networks and Brain Proceedings, Oct. 27-30, 1998, Beijing, China.

[8]. R. Geoff, "Intelligent Mechatronics", Computing and Control Engineering Journal, vol. 9, No. 6, Dec. 1998.