CHAPTER 4

GENETIC ALGORITHM WITH NEURAL NETWORK APPROACH

4.1 INTRODUCTION

Neural networks and Genetic Algorithms demonstrate powerful problem solving ability. They are based on quite simple principles, but take advantage of their mathematical nature: non-linear iteration. Neural networks with backpropagation learning showed results by searching for various kinds of functions. However, the choice of the basic parameter (network topology, learning rate, initial weights) often already determines the success of the training process. Research on using genetic algorithms for neural networks learning is increased. Typically, genetic search is used for the weights optimization on a pre-specified neural network topology. However, determining the appropriate size of a neural network is one of the most difficult tasks in its construction.

To overcome the problems caused by Evolutionary Algorithm on efficiency, population diversity and premature convergence when applying to web service selection problem, a hybrid algorithm has been proposed which combines both the GA and Neural Networks. In recent years the evolutionary approach to artificial neural networks has been an emerging technology to solve combinatorial optimization problems. Here the GA alters the weight attribute values for Neural Networks using the crossover and mutation.

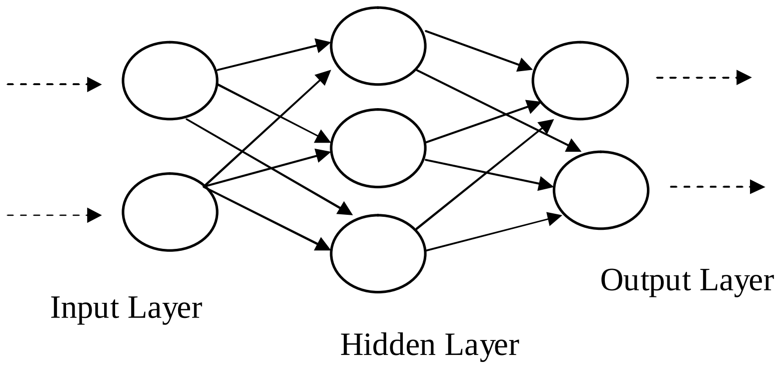
4.2 NEURAL NETWORKS

Neural Networks (NN) finds applications in every field to solve the classification or approximation problem. NN is developed in 1943 and it is an algorithm designed for the learning and optimization to mimic the human brain. A neural network is a computational model consisting of a number of connected elements, known as neurons. A neuron is a processing unit that receives input from outside the network and/or from other neurons, applies a local transformation to that input, and provides a single output signal which is passed on to other neurons and/or outside the network. Each of the inputs is modified by a value associated with the connection. This value is referred to as the connection strength, or weight, and roughly speaking, represents how much importance the neuron attaches to that input source.

Many research works have been done in the past decade by using genetic algorithms to adjust the weights of neural networks. Initially this combination found to be slow and now due to the improvements both the algorithm have gained speed and popularity at finding unknown variables (i.e) the parameters which represent the neural network. This combination proves its performance in predicting sun spots by Koehn (1994).

Basic components of NN are explained as follows. Nodes (Neurons) are connected with each other through links. These links are assigned with weights. Weights are an important part in calculation of input during the training and testing phase. Each node in the network contains an input and activation function. The activation function is always a summing function. Common activation functions are linear activation and sigmoid activation function. An artificial neural network (ANN) contains an input layer, hidden layer and output layer.

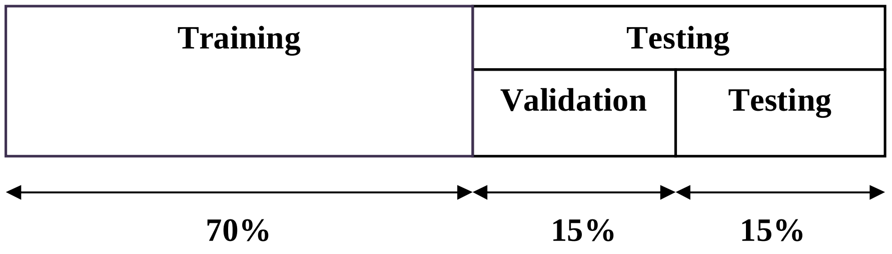
Nodes in the input layer denote the number of parameters present in the dataset we provide, output nodes differ according to the domain. A network without hidden units is called as linear network. Hidden layers are added between the input and output layer to make the network more accurate. The idea behind the neural networks is a chain reaction (i.e) when input layers create output the same will be provided as the input to another input layer and so called Backpropagation by Almeida & Ludermir (2008). Application of NN in health industries is that it is used to predict diabetes. The structure of the NN is given in the Figure 4.1.



(Source: Almeida & Ludermir 2008**)**

Figure 4.1 Structure of a Network

Training phase makes the output of the network to produce the desired output. Training a network can be of many types. A popular way to train the network is by adjusting the weights depending on the error values propagated through other layers. Network should works well even in the unknown situations. Over fitting adjusts the training dataset and loses generalization capability. Hence to overcome this problem the dataset is divided in to two parts: 70% is used for training and remaining 30% for testing. Again the testing data is divided in to 50% of the samples for new training and a validation set consisting of remaining 50% as shown in below Figure 4.2. In the proposed method the fitness of chromosomes are not verified with reference to training but with separate test data set. When this result in error rate below the threshold, training process is interrupted by Lisboa & Taktak (2006).



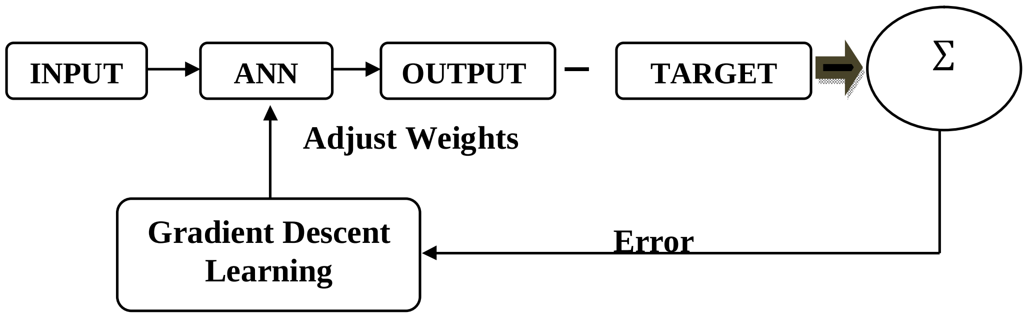
(Source: Lisboa & Taktak 2006**)**

Figure 4.2 Dataset partition for training and testing

Training and learning algorithms are used to adjust the weights. These algorithms include Gradient descent, Hebbian learning, LVQ, Widrow-Hoff, and Kohonen. Out of these, gradient descent learning is the most commonly used algorithm for training a feed forward network.

4.2.1 Training Neural Networks with Gradient Descent Training

The flow chart of ANN with gradient descent learning is shown in the Figure 4.3.



(Source: Lisboa & Taktak 2006**)**

Figure 4.3 Flow chart of ANN with Gradient Descent Learning

A neural network maps the input through iterative adjustment of weights. Information's from inputs are passed to the network to optimize the weights between nodes. The optimization of the weights is done by the back propagation of the errors produced during training phase. The ANN reads the inputs as well as outputs in the training dataset provided and changes the values of weights to reduce the error between the predicted and target values. This process continues until the specified level of accuracy is achieved to minimize the predicted error. A cycle of forward-backward pass and weight adjustments using input and output in dataset is called as epoch or iteration. If the network is over trained it will lose the ability to generalize. Hence, the training or learning method should be chosen suitably.

Learning methods in neural network is of three types Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Most common method of unsupervised learning technique used in feed forward networks is delta rule or backpropogation rule. Normally, output of the nodes is function of its inputs. The inputs to the node are products of output of preceding nodes with their corresponding weights. These are summed and passed through activation function before it is sent out from nodes. This can be written as

$$X_i = \sum_i w_{ij} a_i \tag{4.1}$$

$$a_j = f(X_i) \tag{4.2}$$

Here X_i is the sum of products of weights and outputs obtained from the previous layer *i*, w_{ij} is the weight connecting layer *i* and layer *j*, a_i represent the activations of the nodes from the previous layer *i*, a_j is the activation of the current node and f is the function which represents activation. The error function is given by,

$$E_{TE} = \frac{1}{2}\sum_n (t_{j_n} - a_{j_n})^2 \tag{4.3}$$

Where E_{TE} represent the total error of training patterns and $\frac{1}{2}$ is applied to simplify the derivative of function, n is the output nodes, t_{j_n}represents the target value for the node n in the output layer j and a_{j_n}represents the actual activation node. Error over the entire training pattern is the sum of E_{TE}

$$E = \sum_p E_{TE} \tag{4.4}$$

Where E represents the total error and p represents training patterns. The mean square error is given by

$$\text{MSE}=\frac{1}{2PN}\sum_p \sum_n (t_{j_n} - a_{j_n})^2 \tag{4.5}$$

Here P and N are the total number of training patterns and output nodes. The aim of the gradient descent function is to minimize the MSE (Equation (4.5)). Delta rule is given by

$$\Delta w_{ij_x} = -\varepsilon \frac{\delta E}{\delta w_{ij}} = \varepsilon \delta a_{i_x} \tag{4.6}$$

From the Equation (4.6) it is clear that the change in a weight of a particular node is equal to the product of learning rate epsilon (ε), difference between target and actual activation function, activation of the input node associated the particular weight during calculation. Each iteration will result in a slight reduce of error. Hence more iteration is required to minimize the error. But this gradient descent learning gets trapped in to local minima. GA enables the learning rules to escape from local minima to avoid the backpropogation algorithm to premature convergence.

```
1. Randomly initialize weights for ANN.
2. Repeat the process until
3. For each weight $w_{ij}$ set $\Delta w_{ij}$=0

    • Set input units from training data.
    • Compute value of output units
    • For each weight $\Delta w_{ij}$ set $\Delta w_{ij}$=$\Delta w_{ij}$+$\frac{\delta E}{\delta w_{ij}}$
4. For each weight $w_{ij}$ set $w_{ij}$=$\Delta w_{ij}$+ $\alpha\Delta w_{ij}$
```

(Source: Lisboa and Taktak 2006**)**

Figure 4.4 Algorithm for Traditional ANN

Neural networks usually use inductive learning (which requires examples) while genetic algorithm uses deductive learning (requires objective function). GA is used in neural networks for three major functions Train the weights, Design the structure, Find optimal learning rule.

4.3 GENETIC ALGORITHM

Genetic algorithms are algorithms for optimization and learning based loosely on several features of biological evolution. They require five components:

1. A way of encoding solutions to the problem on chromosomes.

2. An evaluation function that returns a rating for each chromosome given to it.

3. A way of initializing the population of chromosomes.

4. Operators that may be applied to parents when they reproduce to alter their genetic composition. Included might be mutation,

crossover (i.e. recombination of genetic material), and domain-specific operators.

5. Parameter settings for the algorithm, the operators, and so forth.

Genetic algorithms should not have the same problem with scaling as backpropagation. One reason for this is that they generally improve the current best candidate monotonically. They do this by keeping the current best individual as part of their population while they search for better candidates. Secondly, genetic algorithms are generally not bothered by local minima. The mutation and crossover operators can step from a valley across a hill to an even lower valley with no more difficulty than descending directly into a valley by Montana & Davis (1989).

A binary representation of the chromosomes is chosen as it maximizes the number of building blocks per information unit. Earlier attempts of GA training of NN use real value strings, meaning that crossover only takes place in the space between weights. This further means that the only way to separate weights are actually changed is by mutation. Such a strategy might not be taking full advantage of the GA's processing capabilities.

4.3.1 Optimizing Weight Attribute with Genetic Algorithm

As the service runs, the user’s preference will change due to difference reasons. For example the users will be more interested in an expensive service with high performance instead of a cheap one with low performance. Hence there is a need to extend the algorithm to a dynamic process. In order to solve this problem, we should focus on the weight adjust factor. Our purpose is to find an objective and automatic way which can adaptively adjust the weight.

Artificial neural network is a non-linear dynamic system composed by a lot of highly complex, distributing, and parallel information processing units. It can learn from the former experiences through adjusting the connection weights and can use the knowledge learned before. Here we have used Genetic algorithm instead of gradient descent algorithm to overcome the problem of local minima and premature convergence. The flow chart for the GANN is shown in the Figure 4.5.

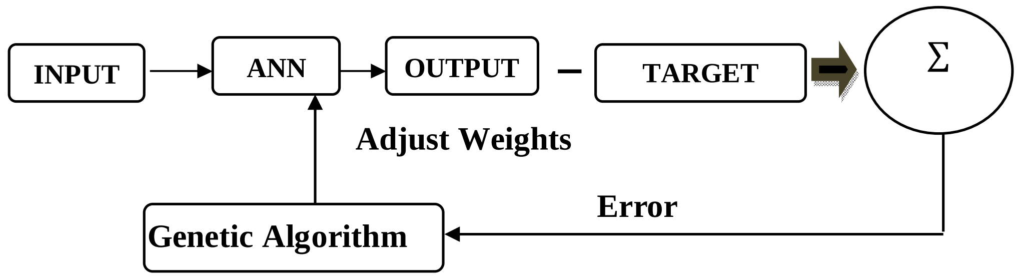


Figure 4.5 Flow chart of ANN with Genetic Algorithm

A cycle of weight adjustments in neural network using GA is done as follows:

Step 1: Decode individuals in the generation in to a set of connection weights and form a corresponding ANN.

Step 2: Compute the mean squared error between actual and target outputs for each ANN. Higher the error lower will be the fitness.

Step 3: Based on the fitness rank select parents for reproduction.

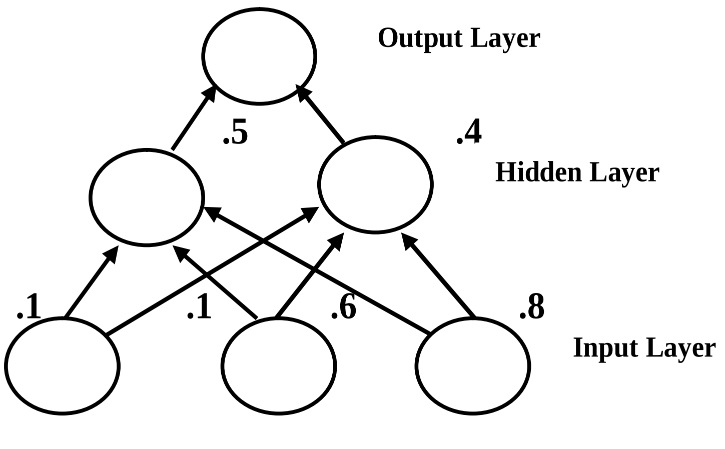
Step 4: Modify the parents with crossover and mutation operators to generate new offspring's for next generation.

Figure 4.6 Algorithm for GANN

An algorithm for the working of GANN is shown in the Figure 4.6. In order to train the system with genetic algorithm first the problem has to be represented as chromosomes as shown in the Figure 4.7. The initial weights are randomly generated which is different from the backpropagation model, since backpropagation uses the probability distribution between -1.0 and 1.0. The focus of our paper is to use real valued genetic algorithm. The fitness functions (see. Equation (4.7)) is defined such that that the performance of the neural network is improved.

$$\text{Fitness Function} = \sum_{i=1}^{N}\left(\sum_{j=1}^{3}\frac{q_{ij}-q_j^{min}}{q_j^{max}-q_j^{min}} + \sum_{i=4}^{5}\frac{q_j^{max}-q_{ij}}{q_j^{max}-q_j^{min}}\right) \qquad (4.7)$$

To evaluate the fitness function each chromosome values are assigned as weight in the links to the network



Encoding: (.5, .4, .1, .8, .3, .1, .7, .6)

Figure 4.7 Encoding of chromosomes

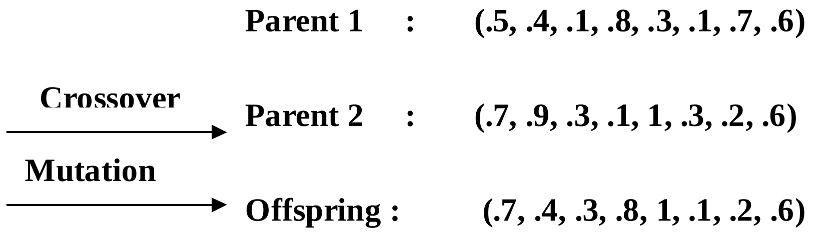

Parent 1 : (.5, .4, .1, .8, .3, .1, .7, .6)
Crossover
Parent 2 : (.7, .9, .3, .1, 1, .3, .2, .6)
Mutation
Offspring : (.7, .4, .3, .8, 1, .1, .2, .6)

Figure 4.8 Operation of Operators

After encoding the chromosomes and performing crossover and mutation operations, the new weights are assigned to the network. The network is then subjected to run with the training samples with the updated weights, this process returns the sum of square of errors. By doing so the genetic algorithm finds a set of optimized weights which minimizes the error function of network. In evolutionary neural learning the genetic algorithm is used to find the optimal set of weights in a network which minimizes the error function.

4.4 EVALUATION RESULTS

We have considered a travel plan dataset which includes Flight reservation, Transportation and Hotel booking services. Quality of Service (QoS) factors for each web services are expressed by the providers. The user may specify their preferences towards quality attributes by assigning weights to them. For example, a person may show interest in choosing a transport at low cost and hotel at higher cost for luxury (Refer Chapter 1). To test the efficiency of genetic algorithm in optimizing the weights in neural network, a comparison between gradient descent, a traditional learning algorithm and genetic algorithm was done. Training phase makes the input of the network to produce the desired output. Training a network can be of many types. A popular way to train the network is by adjusting the weights depending on the error values propagated through other layers. Network should work well even in the unknown situations.

Over fitting adjusts the training dataset and loses generalization capability. Hence to overcome this problem the dataset is divided in to two parts: 70% is used for training and remaining 30% for testing. Again the testing data is divided into 15% of the samples for new training and a validation set consisting of remaining 15%. In the proposed method the fitness of chromosomes are not verified with reference to training but with separate test data set. When this results in an error rate below threshold, the training process is interrupted by Lishoba et al (2006). A 150x5 matrix was formed with 150 web services with four quality attributes that are generated randomly and used as dataset. Out of 150 samples 70% of the samples were taken for training and 15% is for validation and 15% for testing.

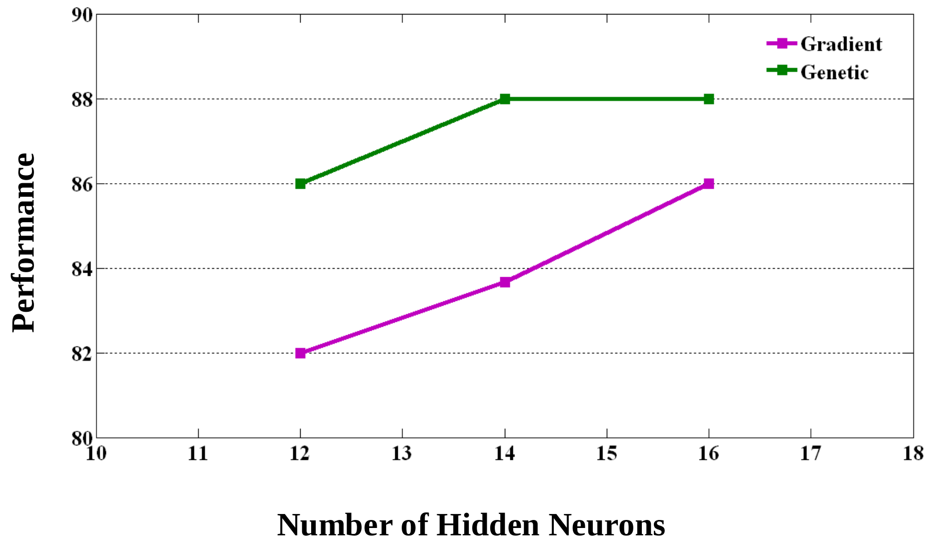


Figure 4.9 Performance comparison between Gradient Descent and Genetic Algorithm learning in Neural Network

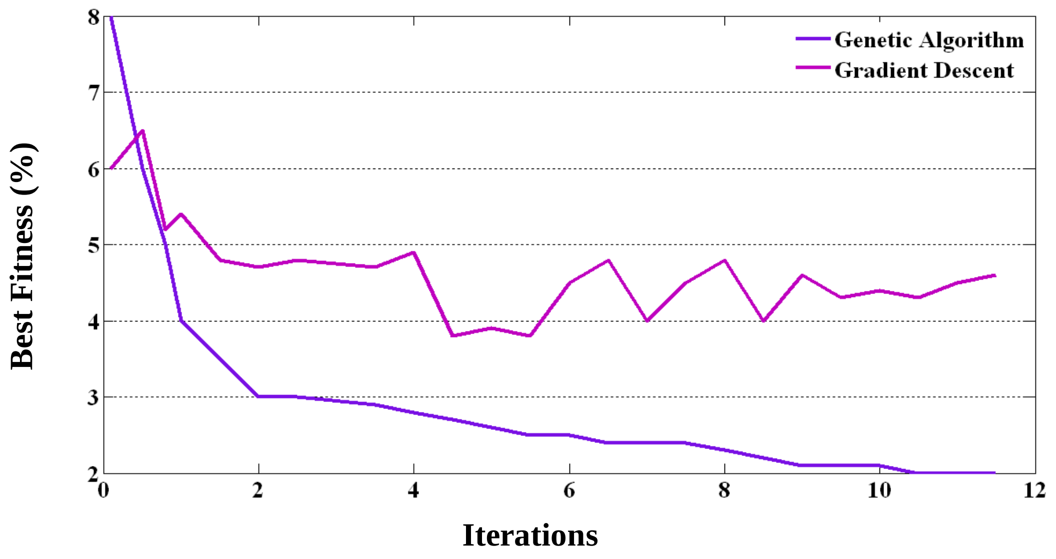


Figure 4.10 Fitness comparison between Gradient Descent and Genetic Algorithm learning in Neural Network

Here we have considered that two iterations of genetic algorithm is equivalent to one cycle of backpropagation. The reason is that back propagation learning makes forward propagation and calculates errors with outputs produced. These errors are then backpropagated and weights are adjusted. Also the execution time for Gradient Descent is more (86 sec) than Genetic Algorithm (72 sec).

4.5 CONCLUSION

The genetic algorithm is proposed to train the neural network for web service selection problem and it is proven to be effective. Both the algorithms have their own strengths and weakness. Back propagation is enough if one adapts fire and forgets mentality towards training. But in some cases where the data sets are more and complicated the genetic algorithm becomes best. Back propagation works well if the data domain has missing values on the other hand if the search space is huge the genetic algorithm yields desirable results.