Initial Classification Through Back Propagation In a Neural Network Following Optimization Through GA to Evaluate the Fitness of an Algorithm

Amit Ganatra[1], Y P Kosta[2], Gaurang Panchal[3], Chintan Gajjar[4]

[1 2 3]Department of Computer Engineering, [4]Information Technology Department
Charotar Institute of Technology (Faculty of Technology and Engineering),
Charotar University of Science and Technology, Changa, Anand-388 421, INDIA

[1]amitganatra.ce@ecchanga.ac.in
[2]ypkosta.adm@ecchanga.ac.in
[3]gaurangpanchal.ce@ecchanga.ac.in
[4]chintangajjar.it@ecchanga.ac.in

Abstract— *an Artificial Neural Network classifier is a nonparametric classifier. It does not need any priori knowledge regarding the statistical distribution of the class in a giver selected data Source. While, neural network can be trained to distinguish the criteria used to classify easily in a generalized manner that allows successful classification the newly arrived inputs not used during training. Through this paper it is eastliblished that back propagation neural network works successfully for the purpose of classification. Back propagation suffers from getting stuck into Local Minima. Weight optimization in Back propagation can be optimized using the Genetic Algorithm (GA). The back propagation algorithm is improved by invoking Genetic algorithm, to improve the overall performance of the classifier. The performance of a fitness algorithm using the approach suggested by us is a Hybrid System that is being analyzed in this paper. In this paper the issue of improving the fitness (weight adjustment) of Back propagation algorithm is addressed. Some of the Advantages of Hybrid algorithms are: convergence speed will be increased and the local minima problem can be overcome. The proposed Hybrid Algorithm is to perform learning as a back propagation and optimize weights using GA for classification.*

Keywords— *Artificial Neural Network, Back propagation algorithm, Genetic algorithm*

I. INTRODUCTION

Classification is a data mining (machine learning) technique used to predict group membership for data instances. Classification means evaluating a function, which assigns a class label to a data item. Classification is a supervised learning process, uses training set which has correct answers (class label attribute).

Classification proceeds as these steps: First create a model by running the algorithm on the training data. Then test the model. If accuracy is low, regenerate the model, after changing features, reconsidering samples. Then identify a class label for the incoming new data. So here the problem is to develop the classification model using the available training set which needs

DOI : 10.5121/ijcsit.2011.3108

to be normalized. Then this data is given to the Back propagation algorithm for classification. After applying Back propagation algorithm, genetic algorithm is applied for weight adjustment. The developed model can then be applied to classify the unknown tuples from the given database and this information may be used by decision maker to make useful decision.

If one can write down a flow chart or a formula that accurately describes the problem, then stick with a traditional programming method. There are many tasks of data mining that are not solved efficiently with simple mathematical formulas.

Large scale data mining applications involving complex decision making can access billions of bytes of data.. Hence, the efficiency of such applications is paramount. Classification is a key data mining technique.

II. INTRODUCTION TO NEURAL NETWORK

Artificial Neural Network (ANN) is a computational model, which is based on Biological Neural Network. Artificial Neural Network is often called as Neural Network (NN). To build artificial neural network, artificial neurons, also called as nodes, are interconnected. The architecture of NN is very important for performing a particular computation. Some neurons are arranged to take inputs from outside environment. These neurons are not connected with each other, so the arrangement of these neurons is in a layer, called as Input layer. All the neurons of input layer are producing some output, which is the input to next layer. The architecture of NN can be of single layer or multilayer. In a single layer Neural Network, only one input layer and one output layer is there, while in multilayer neural network, there can be one or more hidden layer.

An artificial neuron is an abstraction of biological neurons and the basic unit in an ANN. The Artificial Neuron receives one or more inputs and sums them to produce an output. Usually the sums of each node are weighted, and the sum is passed through a function known as an activation or transfer function. The objective here is to develop a data classification algorithm that will be used as a general-purpose classifier. To classify any database first, it is required to train the model. The proposed training algorithm used here is a Hybrid BP-GA. After successful training user can give unlabeled data to classify

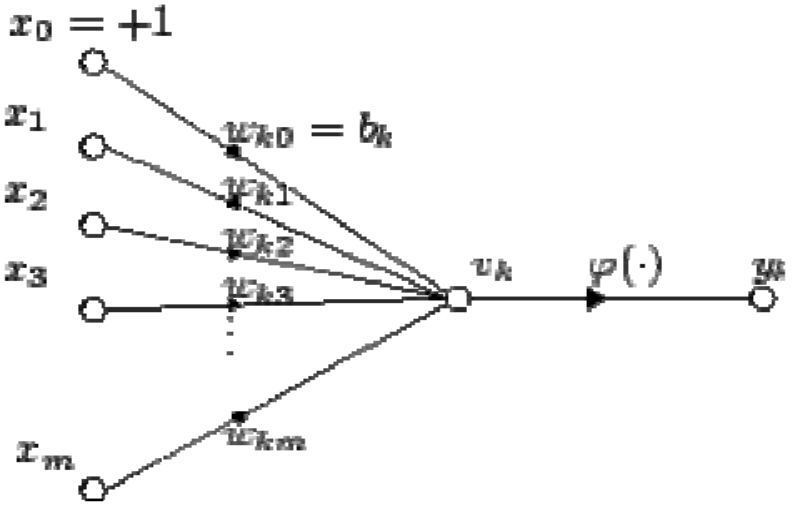

Fig. 1 Artificial Neurons

The synapses or connecting links: that provide weights, wj, to the input values, xj for j = 1 ...m;
An adder: that sums the weighted input values to compute the input to the activation function

$$v = w_0 + \sum_{j=1}^{m} w_j x_j$$

Where,
w0 is called the bias, is a numerical value associated with the neuron. It is convenient to think of the bias as the weight for an input x0 whose value is always equal to one, so that;

$$v = \sum_{j=0}^{m} w_j x_j$$

An activation function g: that maps *v* to *g*(*v*) the output value of the neuron. This function is a monotone function. The logistic (also called the sigmoid) function $g(v) = (e^v/(1+e^v))$ as the activation function works best. The practical value of the logistic function arises from the fact that it is almost linear in the range where g is between 0.1 and 0.9 but has a squashing effect on very small or very large values

III. ANN ARCHITECTURE : MULTILAYER PERCEPTRON

The Perceptron

One neuron can't do much on its own. Usually we will have many neurons labelled by indices k, i, j and activation flows between them via synapses with strengths wki, wij:

One can connect any number of McCulloch-Pitts neurons together in any way one like. An arrangement of one input layer of McCulloch-Pitts neurons feeding forward to one output layer of McCulloch-Pitts neurons is known as a Perceptron. Perceptron is simple neural network consisting of input layer, output layer, and possibly one or more intermediate layers of neurons.

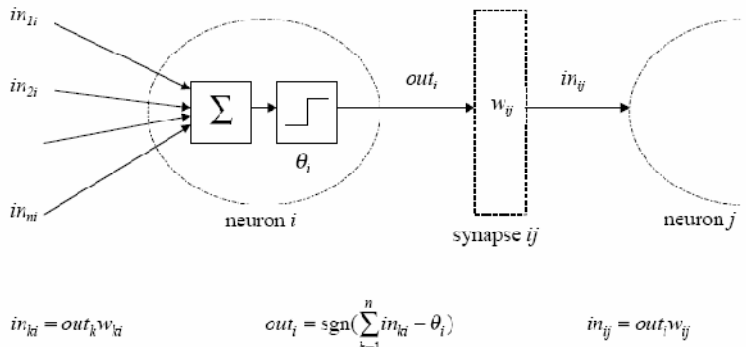

Fig. 2 ANN Architecture: Multilayer Perceptron

MLPs are one of the most common neural network structures, as they are simple and effective, and have found home in a wide assortment of machine learning applications, such as character recognition.

A typical Back Propagation Neural Network (BPNN) start as a network of nodes arranged in three layers--the input, hidden, and output layers. The architecture of BPNN is same as MLP and the learning rule applied is BP algorithm. There is no theoretical limit on number of hidden layers but there are just one or two. The input and output layers serve as nodes to buffer input

and output for the model, respectively, and the hidden layer serves to provide a means for input relations to be represented in the output.

Before any data has been run through the network, the weights for the nodes are random, which has the effect of making the network much like a newborn's brain--developed but without knowledge. When the connections of the network are going forward the architecture is called Feed Forward neural network.

2.1 ANN Learning: Back Propagation Algorithm

The back propagation algorithm cycles through two distinct passes, a forward pass followed by a backward pass through the layers of the network. The algorithm alternates between these passes several times as it scans the training data.

Forward Pass: Computation of outputs of all the neurons in the network

- The algorithm starts with the first hidden layer using as input values the independent variables of a case from the training data set.
- The neuron outputs are computed for all neurons in the first hidden layer by performing the relevant sum and activation function evaluations.
- These outputs are the inputs for neurons in the second hidden layer. Again the relevant sum and activation function calculations are performed to compute the outputs of second layer neurons.

Backward pass: Propagation of error and adjustment of weights

- This phase begins with the computation of error at each neuron in the output layer. A popular error function is the squared difference between ok the output of node k and yk the target value for that node.
- The target value is just 1 for the output node corresponding to the class of the exemplar and zero for other output nodes.
- The new value of the weight wjk of the connection from node j to node k is given by: wnewjk= woldjk+ηojδk. Here η is an important tuning parameter that is chosen by trial and error by repeated runs on the training data. Typical values for η are in the range 0.1 to 0.9.
- The backward propagation of weight adjustments along these lines continues until we reach the input layer.
- At this time we have a new set of weights on which we can make a new forward pass when presented with a training data observation

2.2 Considerations for choosing Network Structure

There are many ways that feed forward neural networks can be constructed. You must decide how many neurons will be inside the input and output layers. You must also decide how many hidden layers you're going to have, as well as how many neurons will be in each of these hidden layers.

The Input Layer:

The input layer to the neural network is the conduit through which the external environment presents a pattern to the neural network. Once a pattern is presented to the input later of the neural network the output layer will produce another pattern. The input layer should represent the condition for which we are training the neural network for.

Every input neuron should represent some independent variable that has an influence over the output of the neural network.

The Output Layer:

The output layer of the neural network is what actually presents a pattern to the external environment. Whatever patter is presented by the output layer can be directly traced back to the input layer. The number of a output neurons should directly related to the type of work that the neural network is to perform.

To consider the number of neurons to use in your output layer you must consider the intended use of the neural network. If the neural network is to be used to classify items into groups, then it is often preferable to have one output neurons for each group that the item is to be assigned into.

If the neural network is to perform noise reduction on a signal then it is likely that the number of input neurons will match the number of output neurons. In this sort of neural network you would one day he would want the patterns to leave the neural network in the same format as they entered.

The Number of Hidden Layers:

There are really two decisions that must be made with regards to the hidden layers. The first is how many hidden layers to actually have in the neural network. Secondly, you must determine how many neurons will be in each of these layers. We will first examine how to determine the number of hidden layers to use with the neural network.

Neural networks with two hidden layers can represent functions with any kind of shape. There is currently no theoretical reason to use neural networks with any more than two hidden layers. Further for many practical problems there's no reason to use any more than one hidden layer.

2.3 Parameters to be considered to build BP algorithm

Initial weight range(r): It is the range usually between [-r, r], weights are initialized between these range.

Number of hidden layers: Up to four hidden layers can be specified; see the overview section for more detail on layers in a neural network (input, hidden and output). Let us specify the number to be 1.

Number of Nodes in Hidden Layer: Specify the number of nodes in each hidden layer. Selecting the number of hidden layers and the number of nodes is largely a matter of trial and error.

Number of Epochs: An epoch is one sweep through all the records in the training set. Increasing this number will likely improve the accuracy of the model, but at the cost of time, and decreasing this number will likely decrease the accuracy, but take less time.

Step size (Learning rate) for gradient descent: This is the multiplying factor for the error correction during back propagation; it is roughly equivalent to the learning rate for the neural network. A low value produces slow but steady learning; a high value produces rapid but erratic learning. Values for the step size typically range from 0.1 to 0.9.

Error tolerance: The error in a particular iteration is back propagated only if it is greater than the error tolerance. Typically error tolerance is a small value in the range 0 to 1.

Hidden layer sigmoid: The output of every hidden node passes through a sigmoid function. Standard sigmoid function is logistic; the range is between 0 and 1.

Output layer sigmoid: Standard sigmoid function is logistic; the range is between 0 and 1.

Critical error: The desired error (MSE) for stopping network training. If the actual error is equal to or less than this error, the training will be terminated.

$$MSE = \sum_{i=1}^{n} (O_i - T_i)^2$$

Where, O is the desired output for training data or cross-validation data i, T is the network output for training data or cross-validation data i,

2.4 Stoping Training Criteria

The epoch/cycle control strategy: where the training will keep going until the training epochs/cycles reach a user defined number.

The error control strategy: when the error of the training set is smaller than a user defined value, training will be stopped. Usually, TSS, MSE, RSME are used.

The proportion control strategy: When the proportion of the number of patterns correctly classified among the number of total training set reaches a pre-defined percentage, the training will be terminated.

The user control strategy: where the user/network trainer forces the training to stop in case he/she thinks there is no need to continue the training.

The early stopping strategy- Validation control: when error on validation set is a minimum. Break the TRAINING set into 2 parts. Use part 1 to compute the weight changes every m epochs apply the partially trained network to part 2 (the validation set) and save the weights.

IV. INTRODUCTION TO GENETIC ALGORITHM

Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Inspired by Darwin's theory of evolution - Genetic Algorithms (GAs) are computer programs which create an environment where populations of data can compete and only the fittest survive, sort of evolution on a computer.

Genetic Algorithms are a search method that can be used for both solving problems and modelling evolutionary systems. Since it is heuristic (it estimates a solution) you won't know if the solution is exact. Most real-life problems are like that: you estimate a solution, you don't calculate it exactly. Most problems don't have any formula for solving the problem because it is either too complex or it takes too long to calculate the solution exactly. One example could be space optimization - the best way to put objects of varying size into a room so they take as little space as possible. A heuristic method is a more feasible approach.

GAs differs from other heuristic methods in several ways. One important difference is that it works on a population of possible solutions, which other heuristic methods use a single solution in their iterations. Another difference is that GAs are probabilistic and not deterministic.

Chromosomes: All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serve as a model for the whole organism. A chromosome consists of genes, blocks of DNA. Each gene encodes a particular protein. Basically, each gene encodes a trait, for example eye color. Possible settings for a trait (e.g.

blue, brown) are called alleles. Each gene has its own position in the chromosome. This position is called locus. A complete set of genetic material (all chromosomes) is called genome. A particular set of genes in genome is called the genotype.

Reproduction: During reproduction, the first thing that occurs is recombination (or cross-over). Genes from the parents combine in some way to create a whole new chromosome. The newly created offspring can then be mutated. Mutation means that the elements of DNA are a bit changed. These changes are mainly caused by errors in copying genes from parents.

Fitness: The fitness of an organism is measured by the success of the organism in its life.

Evolution: Evolution is a process that each and every species in nature is subjected to. In evolution, each species faces the problems of searching for a beneficial adaptation to adapt to the rapidly changing environment around them.

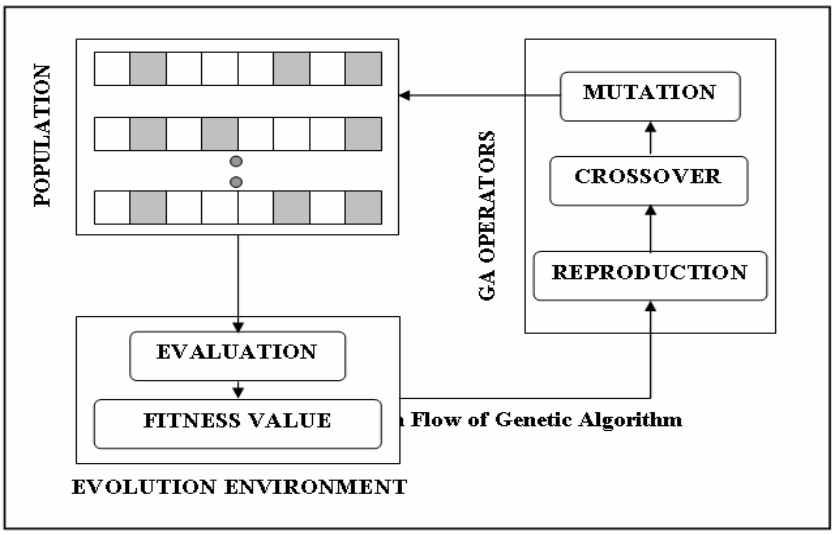

Fig. 3 Outline of Genetic Algorithm

Start with a set of possible solutions (represented by chromosomes) the population. Solutions from one population are taken and used to form a new population. This is motivated by a hope that the new population will be better than the old one. New solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce by mating (crossover). Repeat the cycle until some condition is satisfied. These steps are described as follows:

1. Generate a random population of n chromosomes which are suitable solutions.

2. Establish a method to evaluate the fitness f(x) of each chromosome x in the population

3. Create a new population by repeating the following steps until the new population is complete

- **Selection:** select from the population according to some fitness scheme.
- **Crossover:** New offspring formed by a crossover with the parents.
- **Mutation:** With a mutation probability mutate new offspring at each locus (position in chromosome).

4. Use the newly generated population for a further run of algorithm.

Why to choose Back Propagation Neural Network?

- A study of comparing Feed forward Network, Recurrent Neural Network and Time-delay neural Network shows that highest correct classification rate is achieved by the fully connected feed forward neural network. [9]
- From table 3.1, it can be seen that results obtained from BPNN are better than those obtained from MLC.
- From the obtained results in table 3.3, it can be seen that MLP is having highest classification rate, with reasonable error and time taken to classify is also reasonably less
- From table 3.3, it can be seen that considering some of the performance parameters, BPNN is better than other methods, GA, KNN and MLC.

Comparison of Classification Methods using WEKA:

Relation: weather
Instances: 14
Attributes: 5 Outlook, Temperature, Humidity, Windy, Play, Test mode evaluated: on training data
Output Class: outlook

TABLE I

COMPARISON OF CLASSIFICATION TECHNIQUES

Classifier	Correctly classified Instances	Incorrectly classified Instances	Root mean squared error	Time to build model in seconds
Multilayer Perceptron	85.71%	14.28%	0.254	0.08
SMO	64.28%	35.71%	0.43	0.53
bayes.net	50%	50%	0.438	0.02
NBTree	50%	50%	0.441	0.03
ZeroR	35.71%	64.28%	0.47	0
Filtered Classifier	64.28%	35.71%	0.388	0.02
MultiClass Classifier	57.14%	42.85%	0.417	0.01
Bagging	71.42%	28.57%	0.395	0.02

Why to combine GA with BP

Many studies have shown that ANNs have the capability to learn the underlying mechanics of time series, or, in the case of trading applications, the market dynamics. However, it is often difficult to design good ANNs, because many of the basic principles governing information processing in ANNs are hard to understand, and the complex interactions among network units usually makes engineering techniques like divide and conquer inapplicable.

When complex combinations of performance criteria (such as learning speed, compactness, generalization ability, and noise resistance) are given, and as network applications continue to

grow in size and complexity, the human engineering approach will not work and a more efficient, automated solution will be needed.

When GA is applied to problem solving, the basic premise is that we can create an initial population of individuals representing possible solutions to a problem we are trying to solve. Each of these individual has certain characteristics that make them more or less fit as members of the population.

The fit members will have a higher probability of mating than the less fit members, to produce offspring that have a significant chance of retaining the desirable characteristics of their parents.

Advantages of Genetic Algorithms

- Crossover is a crucial aspect of any genetic algorithm, but it may seem that it will dramatically change parents with a high fitness function so that they will no longer be fit. However this is not the case. As in biology, crossover can lead to new combinations of genes which are more fit than any in the previous generations. Other offspring will be created which are less fit but these will have a low probability of being selected to go on to the next generation.
- Creating new variants is the key to genetic algorithms, as there is a good chance of finding better solutions. This is why mutation is also a necessary part of the program. It will create offspring which would not have arisen otherwise, and may lead to a better solution.
- Other optimization algorithms have the disadvantage that some kind of initial guess is required and this may bias the final result. GAs on the other hand only requires a search range, which need only be constrained by prior knowledge of the physical properties of the system.

Effectively they search the whole of the solution space, without calculating the fitness function at every point. This can help avoid a danger in any optimization problem: being trapped in local maxima or minima.

There are two main reasons for this: i) the initial population, being randomly generated, will sample the whole of the solution space, and not just a small area; ii) variation-inducing tactics, i.e. crossover and mutation, prevent the algorithm being trapped in one part of the solution space.

Types of Hybrid (GA, NN, FL) System:

The types of hybrid systems can be of following types [16].

Sequential Hybrids: Pipeline like architecture. GA pre-processor, which obtains the optimal parameters for different instances of a problem and hands over the 'pre-processed' data set to an NN for the further processing

Auxiliary Hybrids: One technology calls the other as a "Subroutine" to process information needed by it. Neuro-Genetic system in which an NN employs a GA to optimize its structural parameters, i.e. parameters which defines its architecture

Embedded Hybrids: Technologies are integrated. NN-FL hybrid system may have an NN, which receives fuzzy inputs, process it and extracts fuzzy outputs as well.

Uses of GA to combine with BPNN:

Genetic algorithms can be used in conjunction with neural networks in the following four ways.

- They can be used to choose the best inputs to the neural network [18].
- Optimize the neural network parameters (such as the learning rates, number of hidden layer processing elements, etc.)
- Train the actual network weights [19].
- To choose/modify the neural network architecture [17] [19].

V. THE PROPOSED HYBRID ALGORITHM

Name of the algorithm: BPNN_GA. Designed for data classification. BPNN_GA is designed for a system to work as a general purpose classifier.

Input to the algorithm: All Attributes of the data set.

- The input data should be divided into two parts. Training set and Test set.
- The training set should be created to include equal number of patterns for each class.
- The training set should be presented as input to the BPNN.

Output of the algorithm: The class label will be predicted for given input pattern.

- There will be one neuron in output layer to predict a class.
- Output of a neuron will predict whether given pattern is classified under that class or not.

Methodology applied in BPNN_GA:

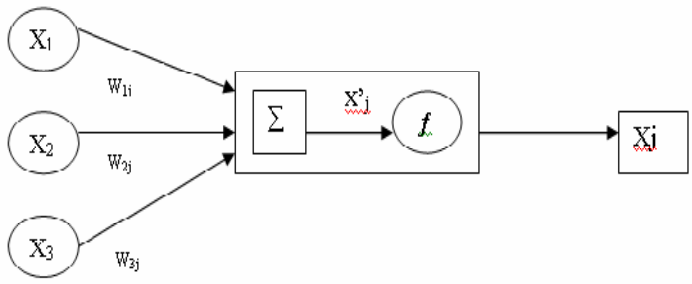

Fig. 4 Methodology Applied in BPNN_GA

Steps to be followed by the proposed system

Step 1: Set Parameters for BPNN_GA algorithm.

Followings parameters are to be set.

- Initial weight range(r):
  The random range [-r, r] of the initial weights.
  Here minimum initial range [-1,1] is taken.
- Critical error:
  The desired error (MSE) for stopping network training

$$MSE = \sum_{i=1}^{n} (O_i - T_i)^2$$

Where, O is the desired output for training data i,
T is the network output for training data i,
If the actual error is equal to or less than this error, the training will be terminated.

• Number of output nodes:

It will be equal to the number of object classes.
In the case of only two output classes, single output neuron is there in output layer.

• Number of input nodes:

For specific feature inputs number of input nodes will be equal to the number of features, say m.
For m attribute data set number of input nodes will be m.

• Number of hidden layers/nodes:

This is critical to decide. In the proposed algorithm, initially 1 Hidden Layer will be considered. Number of nodes 'n' in Hidden layer will be problem dependent.
Initially 'n' is taken minimum

- **Activation function:**

Sigmoid Function f() is chosen as an activation function for Hidden layer neurons and output layer neurons

$$f(x) = 1 /(1+\exp(-x))$$

For input layer, activation function of each neuron is a linear Function, so output is same as input given to the neurons.

- **Learning rate:**

In the proposed system, initially learning rate should be taken Small, in the range of [0.5 - 0.8]

- **Error Tolerance:**

It is the tolerable error, Etol, up to that the difference between Obtained output and desired output is tolerable.

For the proposed system Etol is 0.01. Again it can be changed depending on application of classification.

- **Number of epochs:**

It is the one sweep through all the records in the training set.
For the proposed algorithm it is kept in range of 300 - 500.

- **Population size :**

It is the parameter that indicates how many chromosomes will be included in one generation. For proposed algorithm population size is kept between 20 - 30.

- **Number of generations Ngen:**

This parameter will be decided according to application. It should range between 200 and 800.

- **Chromosome encoding type:**

For this system, initial weights taken are real numbers, so the chromosome will also be encoded as real values.

- **Crossover type:**

  For the proposed algorithm it is taken as 1 point crossover.

- **Crossover rate:**

  It is application dependent. Normally crossover rate ranges between 0.5 - 0.8.

- **Mutation type:**

  In the proposed algorithm, flip the gene value with a random value within range of [-1, 1], which is same as the range considered for assigning initial weights.

- **Mutation rate:**

  In the proposed algorithm mutation rate is kept 0.01.

Step 2: Apply training pattern to BPNN

- Give input values to the BPNN from training patterns.
- Calculate output of each layer according to activation function described in the parameter list above.

Step 3: Calculate error between in obtained output and desired output.

Once the output of output layer is calculated, find error.

$$Ep = \Sigma_k (d_k - x_k)^2$$

Where,

dk is desired output for node k
xk Actual output for node k
Ep is the Mean Square error when pth pattern is given as input

Increment epoch count

If Ep <= Etol OR Epoch count >=300 then Save Δwki and go to step 7, else continue.

Step 4: Calculate Error Gradient vector for node i

By the chain rule, the recursive formula for εi can be rewritten as [21],

$$\varepsilon i = \delta Ep / \delta x'_i$$

$$\varepsilon i = -2(di - x_i)\, x_i(1 - xi)$$; If node i is a output node

$$\varepsilon i = x_i(1 - xi)\, \Sigma j, i<j\ \varepsilon j .\ w_{ij}$$; Otherwise

Step 5: Adjust the weights using error gradient after each epoch.

$\Delta wki = -\eta\ \delta Ep/\delta\ wki$, so that

$$\Delta w_{ki} = -\ \eta\ \varepsilon i x_i$$

Where,
Δwki is a changed weight for connection between node k and i.
n is the number of data in the training data set
Ep is the Mean Square error when pth pattern is given as input.

Go to step 3.

Step 6: Create a chromosome.

Create initial chromosome from the saved weight matrix Δwki. This chromosome should represent the weights of BPNN at the last epoch.

Step 7: Create a population.

Randomly create n-1 chromosomes in the range of [-1, 1], for population size of n. All the chromosomes are created using real value encoding.

Step 8: Apply training pattern and get output.

Apply training patterns from training set and get out
Calculate output of each layer according to activation function

Described in the parameter list above

Step 9: Calculate fitness for each chromosome in a population.

Here fitness function taken is same as the equation used in step 4.

$$Ep = \Sigma_k\ (d_k - x_k)^2 / n$$

If Ep<= Etol OR generation count >= Ngen then stop.

Else continue.

Step 10: Discard percentage of population which is the least fit.

Perform selection according to fitness value of chromosomes.

Step 11: Perform crossover and mutation.

Perform single point crossover, each chromosome is randomly paired and combined with another to create two offspring chromosomes and then random mutation to create a new population. Increment generation count

Pseudo code for Create Network

Input: Number of layers, learning rate

Output: New Network created
Method:
1. Count initial number of layers number of Layers
    Network.LayerCount = UBound(ArrayOfLayers)

```
2. Check if no of layers are not enough
    If Network.LayerCount < 2 Then
    CreateNet = 0
        Exit Function
    End If
3. Initialize all neurons

Set the bias to random value
Network.Layers(i).Neurons(j).Bias = GetRand
Doesn't initialize dendrites for output layer because output layers doesn't have any dendrites
Network.Layers (i).Neurons(j).DendriteCount = ArrayOfLayers(i - 1)
Set the weight of each dendrite
For k = 1 To ArrayOfLayers(i - 1)
DoEvents
Network.Layers(i).Neurons(j).Dendrites(k).Weight = GetRand
Next k
```

Pseudo code to Run Network

```
Input: Number of layers in a network, number of neurons in each layer
Output: Values calculated for each neuron
Method:
1. Check whether number of neurons in input layer are equal to number of input attributes
    If UBound(ArrayOfInputs) <> Network.Layers(1).NeuronCount Then
    Run = 0
    Exit Function
    End If
2. For each layer in the BPNN
            For each neuron in a layer
            Calculate weighted sum of inputs
            Calculate activation as a sigmoid function
            Next
        Next
For i = 1 To Network.LayerCount
        For j = 1 To Network.Layers(i).NeuronCount
        Network.Layers(i).Neurons(j).Value = 0 'First set the value to zero
For k = 1 To Network.Layers(i - 1).NeuronCount
            Network.Layers(i).Neurons(j).Value = Network.Layers(i).Neurons(j).Value +
                Network.Layers(i - 1).Neurons(k).Value *
                Network.Layers(i).Neurons(j).Dendrites(k).Weight
Next k
            Network.Layers(i).Neurons(j).Value =
            Activation(Network.Layers(i).Neurons(j).Value +
            Network.Layers(i).Neurons(j).Bias)
Next j
Next i
    3. Assign values to output layer
            For i = 1 To (Network.Layers(Network.LayerCount).NeuronCount)
            OutputResult(i) = (Network.Layers(Network.LayerCount).Neurons(i).Value)
            Next i
```

Pseudo code to Train Network
Input: Number of layers in a network, number of neurons in each layer
Output: input data, output data
Method:

```
1. Check if correct amount of input is given
    If UBound(inputdata) <> Network.Layers(1).NeuronCount Then
    Train = 0
    Exit Function
    End If
2. Check if correct amount of output is given

    If UBound(outputdata) <> Network.Layers(Network.LayerCount).NeuronCount Then
    Train = 0
    Exit Function
    End If
3. Calculate values of all neurons and set the input

    call Run (inputdata)
4. Calculate Delta for each layer
    For i = 1 To Network.Layers(Network.LayerCount).NeuronCount
          Network.Layers(Network.LayerCount).Neurons(i).Delta =
          Network.Layers(Network.LayerCount).Neurons(i).Value * (1 -
          Network.Layers(Network.LayerCount).Neurons(i).
          Value) * (outputdata(i) - Network.Layers(Network.LayerCount).Neurons(i).Value)
          For j = Network.LayerCount - 1 To 2 Step -1
          For k = 1 To Network.Layers(j).NeuronCount
          Network.Layers(j).Neurons(k).Delta = Network.Layers(j).Neurons(k).Value * (1 -
          Network.Layers(j).Neurons(k).Value) * Network.Layers(j +
          1).Neurons(i).Dendrites(k).Weight * Network.Layers(j + 1).Neurons(i).Delta
    Next k
        Next j
                Next i
5. Calculate new Bias
6. Calculate new Weights
```

VI. DATASET USED FOR CLASSIFICATION PERFORMANCE ANALSYSIS

In this section data sets are taken from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/nurcery. Performance analysis of the system is considered based on below data sets.

TABLE II
DATASET TAKE FOR ANALYSIS

Atttribute Name	Attribute values
Parents	Usual, pretentious, great_prêt.
has_nurs	proper, less_proper, improper, critical, very_crit.
Form	Complete, completed, incomplete, foster.

Children	1, 2, 3, more.
Housing	Convenient, less_conv, critical.
Finance	Convenient, inconv.
Social	Nonprob, slightly_prob, problematic.
Health	Recommended, priority, not_recom.
Classes	Not_recom, recommend, very_recom, priority, spec_prior

Number of Instances in Training set : 7494
Number of Instances in testing set : 3498
Number of Attributes: 16 input +1 class attribute
For Each Attribute:
All input attributes are integers in the range 0...100.
The output attribute is the class code 0...9

VII. PERFORMANCE ANALYSIS

Performance analysis based on Iteration

The Above table shows the performance analysis of neural network using different learning rate.

Performance analysis based on Iterations

TABLE III
PERFORMANCE ANALYSIS BASED ON ITERATIONS

Iterations	Accuracy %	Time (Sec)
100	80	17
200	81	34 sec
300	83	53
500	83	1.23
1000	84	2.37

Performance analysis based on number of Hidden nodes

Other parameters are Learning Rate =0.4, Error Tolerance =80%, Iterations =300, No of Hidden layers = 1, No of Hidden Nodes = 10, Mutation Rate = 0.2, Cross Over Rate = 0.3, Generation = 25, Population = 300

TABLE VI
PERFORMANCE ANALYSIS BASED ON HIDDEN NODES

Hidden Nodes	Accuracy %	Time In sec
8	73	40
9	84	45
10	80	46
11	81	50

Performance analysis based on number of Hidden layers

TABLE VII
PERFORMANCE ANALYSIS BASED ON HIDDEN LAYERS

Hidden Nodes in layer1	Hidden Nodes in layer 2	Accuracy %	Time in sec
6	3	82	26
7	2	74	29
8	1	39	27
6	4	83	28
5	5	81	27

The Above table shows the performance analysis at various hidden Layers. It is vary difficult to select number of hidden layer while selecting architecture. So here we have tested some neural network with different hidden layers and hidden neurons

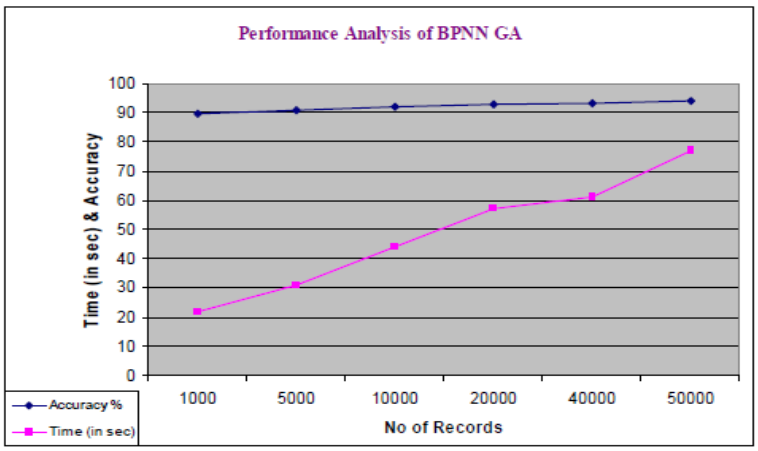

Fig. 5 Performance Comparison of Classification Techniques

Above figure shows the performance analysis of Back Propagation Neural Network and Genetic Algorithm.

Comparing performance with other classification system

Comparing the proposed system with MLP training, Naïve Bayes and some other were done using a tool WEKA. Data set taken for testing is Nursery Data set, having # variants 12120 and training set is having no of variants 12960.

1. Naïve Bayes
2. Bayesian network
3. Support Vector Machine

Naive Bayes Algorithm: The Naive Bayes (NB) algorithm makes predictions using Bayes Theorem, which derives the probability of a prediction from the underlying evidence. NB affords fast model build and apply. NB assumes that each attribute, or piece of evidence, is independent from the others. In practice, this assumption usually does not degrade the model's predictive accuracy significantly

Bayesian Network: It is a probabilistic graphical model that represents a set of variables and their probabilistic independencies. A framework for representing uncertainty in our knowledge. A Graphical modelling framework of causality and influence. A Representation of the

dependencies among random variables. A compact representation of a joint probability of variables on the basis of the concept of conditional independence

Support Vector Machine Algorithm (SVM): Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over fit of the data. Neural networks and radial basis functions, both popular data mining techniques, can be viewed as special cases of SVMs. SVMs perform well with real-world applications such as classifying text, recognizing hand-written characters, classifying images. There is no upper limit on the number of attributes and target cardinality for SVMs. The SVM kernel functions currently supported are linear and Gaussian.

TABLE VIII
COMPARISON OF CLASSIFICATION TECHNIQUES

Algorithm Name	Accuracy
Bayesian network	0.803047372
Naive Bayes	0.80386569
Support Vector Machine	0.815737546
BPNN GA	0.9222595

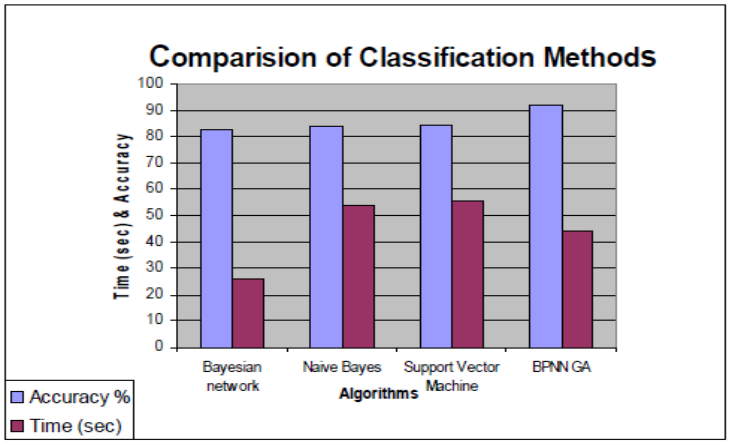

Fig. 6 Performance Comparison of Classification Techniques

Above table and figure shows the accuracy measured of various Classification Methods also the graph shows the Best performance of Back Prorogation Neural Network with Genetic Algorithm better as compare to all other Techniques.

VIII CONCLUSION

Both Genetic Algorithm and back propagation are search and optimization techniques. The goal of this hybrid algorithm is to perform weight adjustment in order to minimize the Mean Square Error between obtained output and desired output. It is better to apply back propagation algorithm first, so that the search space of Genetic algorithm will be reduced. Hence one can overcome the problem of local minima. The proposed algorithm exploits the optimization

advantages of GA for the purpose of accelerating neural network training. BP algorithm is sensitive to initial parameters and GA is not. BP algorithm has high convergence speed and while GA is having slow convergence. So the proposed system blends merits of both BP and GA.

IX FUTURE WORK

In the proposed system, it is assumed that all the input data set has no noise and missing values because artificial neural networks have high tolerance to noisy data. So missing value handling along with continues & categorical values handling should be done as future enhancement.

X ACKNOWLEDGEMENT

The authors' wishes to all the colleagues for their guidance, Encouragement and support in undertaking the research work. Special thanks to the Management for their moral support and continuous encouragement.

IX.REFERENCES

[1] S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2] J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[4] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[6] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[7] M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[8] *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[9] "PDCA12-70 data sheet," Opto Speed SA, Mezzovico, Switzerland.

[10] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[11] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[12] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.