A Genetic Algorithm based Back Propagation Neural Network for Weather Forecasting

(ANN Course Project)

Lilly Kumari
lilly.k0501@gmail.com.
Roll No:12214012

Mehak Gupta
mehakgupta210@gmail.com
Roll No: 12118046

Abstract

Precise weather forecasting is important in today's world considering that the agricultural and industrial sectors are heavily dependent on the weather conditions. Also, forecasting warns us about natural disasters. Because of the non-linearity in climatic physics, neural networks are suitable to predict these meteorological processes. Back Propagation algorithm using gradient descent method is the most important algorithm to train a neural network for weather forecasting. Back propagation algorithm suffers from several problems like getting stuck in local minima. In this paper, in order to overcome some of these problems, an integrated back propagation based genetic algorithm technique to train artificial neural networks is proposed. In the proposed technique, back propagation is combined with genetic algorithm in such a way that the pitfalls of the algorithm get converted to benefits. The results and comparison of the technique with the previous one are enlisted along with.

1. Main Objectives

- Use genetic algorithm for updation of weights in back propagation.
- Train the model with weather forecasting data and compare.

2. Introduction and Background

Forecasting is a phenomenon of knowing what may happen to a system in the next coming time The parameters required to predict weather are enormously complex such that there is uncertainty in prediction even for a short period. Inspired by the brain, neural networks are an interconnected network of processing elements called neurons. Neural networks learn by example i.e. they can be trained with known examples [7]. One of the most popular training algorithms in the domain of neural networks used so far, for weather forecasting is the back propagation algorithm. It is a gradient descent method. The algorithm suffers from many problems. Several attempts have been made by various researchers to solve these problems using genetic algorithms, the computerized search and optimization algorithms that mimic the principle of natural genetics and natural selection [5], [6]. But, in the field of weather forecasting, such efforts are still to be put up. So, the motivation of this work is firstly, to integrate BPN with GA in such a way that the disadvantages of back propagation algorithm get converted to benefits, used further for an accurate weather forecasting model and secondly, to perform comparison between the two main techniques- back propagation network based on gradient descent technique and back propagation network based on genetic algorithm to train the neural network. Language used for coding is C in Visual Studio. The remainder of the article is organized as follows. Section 2 and Section 3 introduce the back propagation algorithm and genetic algorithms respectively. The details of the integrated BP/GA technique for weather forecasting model are shown in Section 4, followed by results in Section 5. Finally, conclusions are summarized in Section 6.

3. Backpropagation Neural Network Training

Initialize all weights in network;
While terminating condition is not satisfied
{ for each training sample X in sample

{ // propagate the inputs forward:
for each hidden or output layer unit
j { Ij=Σj WijOi ; //Compute the net input of unit j with respect to the previous layer, i
$Oj=1/(1+e^{-Ij})$;}
//Compute the output of each unit j
//Backpropagate the errors
for each unit j in the output layer
Errj=Oj (1-Oj) (Tj – Oj);//Compute the error for each unit j in the hidden layers,from the last to the first hidden layer
Errj=Oj (1-Oj) Σk ErrkWjk ; //Compute the error with respect to the next higher layer,k
for each weight Wij in network
{ ΔWij=(l)Err jOi
//Weight increment
Wij=Wij+ΔWij;}//weight update } }
The weights in the network are initialized to small random numbers (e.g., ranging from – 0.0 to 1.0).Then propagate the inputs forward, the input and output of each unit in the hidden and output layers are computed. First, the training sample is fed to the input layer of the network. For unit j in the input layer, its output is equal to its input, that is, Oj= Ij for input unit j. The net input to each unit in the hidden and output layers is computed as a linear combination of its inputs. The inputs to the unit are, in fact, the outputs of the units connected to it in the previous layer. To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight, and this is summed. Given a unit j in a hidden or output layer, the net input, I_j, to unit j is $I_j= \Sigma_i W_{ij} O_i$, where wij is the weight of the connection from unit i in the previous layer to unit j;
Oi is the output of unit I from the previous layer. Each unit in the hidden and output layers takes its net input and then applies an activation function to it. The function symbolizes the activation of the neuron represented by the unit. The logistic or sigmoid function is used. Given the net input Ij to unit j, then Oj, the output of unit j,is computed as $Oj=1/(1+e^{-Ij})$.This function is also referred to as a squashing function[8], since it maps a large input domain onto the smaller range of 0 to 1. The error is then propagated backwards by updating the weights to reflect the error of the network's prediction. For a unit j in the output layer, the error Err_j is computed by $Err_j=Oj(1-Oj)(Tj-Oj)$, where Oj is the actual output of unit j, and Tj is the true output, based on the known class label of the given training sample. Oj(1-Oj) is the derivative of the logistic function. To compute the error of a hidden layer unit j, the weighted sum of the errors of the units connected to unit j in the next layer is considered. The error of a hidden layer unit j is $Err_j = Oj (1-Oj) \Sigma_k Err_k W_{jk}$,where W_{jk} is the weight of the connection from unit j to a unit k in the next higher layer, and Err_k is the error of unit k. The weights are updated to reflect the propagated errors. Weights are updated by the following equations, where Δwij is the change in weight wij: Δwij=(l) ErrjOi
wi j= wij+ Δwij. The variable l is the learning rate ,a constant typically having a value between 0.0 and 1.0.
Training stops when
• all Δwij in the previous epoch were so small
as to below some specified threshold, or
• the percentage of samples misclassified in the previous epoch is below some threshold, or
• a prespecified number of epochs has expired.

4. Genetic Algorithm.

A genetic algorithm is an iterative procedure maintaining a population of structures that are candidate solutions to specific domain challenges [9]. During each temporal increment (called a generation), the structures in the current population are rated for their effectiveness as mutation. Genetic Algorithms (GAs) are search algorithms based on the mechanics of the natural selection process (biological evolution). The most basic concept is that the strong tend to adapt and survive while the weak tend to die out. That is, optimization is based on evolution, and the "Survival of the fittest"

concept. GAs has the ability to create an initial population of feasible solutions, and then recombine them in a way to guide their search to only the most promising areas of the state space. Each feasible solution is encoded as a chromosome (string) also called a genotype, and each chromosome is given a measure of fitness via a fitness (evaluation or objective) function. The fitness of a chromosome determines its ability to survive and produce offspring. A finite population of chromosomes is maintained. GAs use probabilistic rules to evolve a population from one generation to the next. The generations of the new solutions are developed by genetic recombination operators

Biased Reproduction: selecting the fittest to reproduce

Crossover: combining parent chromosomes to produce children chromosomes

Mutation: altering some genes in a chromosome.

Code for mutation:-

```
void mutate(int **mp)
{
int i,j,k,x[6],y;
float z;
for(i=1;i<=p;i++)
{
y = rand()%100;
z = y/100.00;
if(z<=pm)
{
for(j=1;j<=nw;j++)
{
for(k=1;k<=5;k++)
{
x[k] = mp[i][j]/pow(10,5-k);
mp[i][j] = mp[i][j]%(int)pow(10,5-k);
y = rand()%100;

z = y/100.00;
if(z<=mr)
{
y = rand()%10;
x[k] = y;
}
}
mp[i][j]=0;
for(k=4;k>=0;k--)
{
mp[i][j]+=x[5-k]*pow(10,k);
}
}
}
}
}
```

Crossover combines the "fittest" chromosomes and passes superior genes to the next generation. Mutation ensures the entire state-space will be searched, (given enough time) and can lead the population out of a local minima. Determining the size of the population is a crucial factor. Choosing a population size too small increases the risk of converging prematurely to a local minimum, since the population does not have enough genetic material to sufficiently cover the problem space. A larger population has a greater chance of finding the global optimum at the expense of more CPU time. The population size remains constant from generation to generation. Fitness Function Drives the Population toward better solutions and is the most important part of the algorithm.

5. Integrated Back Propagation based Genetic Algorithm (BP/GA Technique)

The proposed weather forecasting model based on BP/GA technique starts with the collection of weather related data, selecting the weather parameters to be forecasted, formation of training data set and testing data set. Finally the methodology and its simulation are provided.

A. Weather Parameters

Sr. no.	Parameters	Unit
1	Air Temperature	ºC
2	Soil Temperature	ºC
3	Relative Humidity	%
4	Vapor pressure	mm
5	Wind speed	Km/ h
6	Wind direction	--
7	Sunshine	hrs
8	Rainfall	mm
9	Evaporation	mm

Table 1.

The daily weather parameters collected from PAU Ludhiana are shown in Table I. along with their units of measurement. The parameters chosen for prediction in this setup are mean air temperature (ºC), relative humidity (%) and daily rainfall (mm). There is no particular reason behind this choice of weather parameters. The choice is made just to predict three main weather variables.

B. Research Data

The data used in this research are the daily weather data for the Ludhiana city of Punjab (India). The data in the unnormalized form have been collected from the "Meteorological Department of Punjab Agriculture University, Ludhiana (Punjab)" of the year 2009. Thirty days data (month of January, 2009) have been used in this research. First twenty five days data have been used for training and next five days data have been used for testing purposes.

C. Normalization of Data

After the collection of data and selection of the weather parameters, next issue is normalization of data. Neural networks generally provide improved performance with normalized data. The use of original data to train the neural network may cause convergence problem. All the weather data sets were, therefore, transformed into values between 0 and 1 through dividing the difference of actual and minimum values by the difference of maximum and minimum values [8].

D. Methodology

Chromosomes form the initial population which is randomly generated and consist of some number of genes. Every gene is encoded of a randomly chosen string length. A particular number of weights are extracted from each chromosome depending upon the number of genes a chromosome have [1]. This calculation is done as follows:

Let the network configuration of the network is l-m-n. Therefore, the numbers of weights (genes), W that are to be determined are:

$W = (l + n) * m$ (1)

With each gene being a real number, and taking the gene length as d, the string representing the chromosomes of weights will have a length of:

$C = W * d$ (2)

It represents the weight matrices of the input-hidden-output layers. With this weight set, the network is trained for the first cycle and the cumulative error is calculated over the inputs obtained from weather forecasting data.

The search for selecting an individual is guided by the fitness of each individual i.e. evaluating the quality of each chromosome [9]. So, the fitness function is evaluated by reciprocating the cumulative error value as follows.

$F = 1/MSE$ (3)

More is the fitness value; more are the chances of a chromosome to be selected for reproducing an offspring. With this criterion in mind, mating pool is prepared by replacing the individual with minimum fitness value by individual having maximum fitness value. For cross over, a two-point cross over method will be used to prepare the new population and the network is considered to be trained when 95% of the individuals have same fitness value [1], [13]. The various steps of weather forecasting model based on BP/GA algorithm are explained below and are shown in fig.1:

1. Start: Generate random population of 'p' chromosomes (suitable solutions for the problem).
2. Extraction: Extract weights for input-hidden-output (lm- n) layers from each chromosome x.
3. Fitness: Evaluate the fitness f(x) of each chromosome x in the population by reciprocating the cumulative error values obtained for each input set (weather forecasting data).
4. New population: Create a new population by repeating following steps until the new population is complete
• Selection: Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
• Crossover: Cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
• Mutation: With a mutation probability mutate new offspring at each position in chromosome.
• Acceptance: Place the new offspring in the new population.
5. Repeat: Repeat steps 3 to 5 until stopping condition is met.

Code for checking when to stop:-

```
int checkstop(int **c,float *f,int *fmax)
{
int maxt=0,ctr,i,j,maxi;
for(i=1;i<=p;i++)
{
ctr=0;
for(j=1;j<=p;j++)
{
if(f[j]==f[i])
{
ctr++;
}
}
if(ctr>maxt)
{
maxt=ctr;
maxi=i;
}
}
for(i=1;i<=nw;i++)
{
fmax[i] = c[maxi][i];
}
if(maxt/p>=0.95)
{
return(1);
}
else
{
return(0);
}
}
```

6. Test: Return the best solution in current population using the test set inputs (weather forecasting data) and the weights (obtained in the above five steps).

E. Simulation

In this research, 3-2-3 neural network architecture has been used. The number of input neurons is 3 representing the date fields, the number of hidden neurons is 2 for processing and the number of outputs is 3 representing the weather variables to be forecasted. A real coding system has been adopted for coding the chromosomes. As the network configuration is 3-2-3, therefore, the numbers of weights (genes) to be determined are 12, as in (1). Taking the gene length as 5, the string representing the chromosomes of weights will have a length of 60, as in (2). This is the weight matrix of the inputhidden-output layers. For cross over, we have used a twopoint cross over selected at random and the selection is made on the basis of fitness value, as in (3). The stopping criterion for the training is when fitness converges.

6. Results and Discussion

The BP/GA technique has been implemented by taking different population sizes. For each value of population, the program has been executed and the error has been calculated.

The error values corresponding to mean air temperature are shown in Table II for the last five days of January 2009.

Table II
Temperature for BP/GA

Day	Desired output	Forecasted Output	Error Value
1	96	97.8	-1.8
2	97	97.9	-0.9
3	98	98.7	-0.7
4	99	99	0.0
5	100	99.3	0.7

Table III shows the prediction of relative humidity along with the error values.

Day	Desired output	Forecasted Output	Error Value
1	1.9	2.1	-0.2
2	2.1	2.4	-0.3
3	2.3	2.3	0.0
4	2.7	2.6	0.1
5	3.0	2.7	0.3

Table III
Relative Humidity for BP/GA

The error values corresponding to daily rainfall parameter is shown in table IV along with the desired output and the forecasted output for the BP/GA technique.

Day	Desired output	Forecasted Output	Error Value
1	15.0	16.6	-1.6
2	16.0	17.1	-1.1
3	17.0	17.8	-0.8
4	18.0	18.6	-0.6
5	19.0	19.0	0.0

Table IV
Rainfall for BP/GA

The above comparison shows clearly that the integrated BP/GA technique is more suitable to predict weather than the traditional gradient based back propagation algorithm because in all cases- temperature, humidity and rainfall; proposed BP/GA technique is more close to the desired output than the back propagation algorithm.

7. Conclusion

The BP/GA technique proposed here can learn efficiently by combining the power of both Genetic Algorithms and Back Propagation. The methodology suggested here is more qualified for neural networks while training them for weather forecasting data considering only global search. Instead of a single point, it works with a population of points taken as instances. It combines the strengths of both deterministic gradient based algorithm and stochastic optimizing algorithm. The BP/GA based on local gradient algorithm is more speed-efficient than the genetic algorithm just overcoming the weakness of agenetic algorithms. Thus the integrated BP/GA technique's use in weather forecasting is encouraged.

References

[1] Rajasekaran S, Vijayalakshmi P., Neural networks, Fuzzy Logic and Genetic Algorithms", New Delhi: Prentice Hall of India, 2004.

[2] David J. Montana and Lawrence Davis, "Training Feedforward Neural Networks Using Genetic Algorithms", BBN Syst. and Tech. Corp. 10 Mouiton St. Cambridge, MA 02138

[3] Azadeh A., et al., "Integration of ANN and GA to Predict Electrical Energy consumption", Dept. of Industrial Eng. and Research Inst. Of Energy Manag. and Planning, Univ. Tehran Iran Islamic Azad Univ., IEEE 2552, 2006.

[4] R. Rojas "Neural Networks- The Backpropagation Algorithm", Springer-Verlag, Berlin, 152, 1996.

[5] Goldberg, D. E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison - Wesley. Reading, MA, 1989.

[6] Holland, J. H., "Adaptation in Natural and Artificial Systems", University of Michigan Press. Ann Arbor, MA, 1975.

[7] Schalkoff, R.J., "Artificial Neural Networks", New York: McGraw-Hill, 1997.

[8] Maqsood I., Khan M.R. and Abraham A., "Weather Forecasting Models Using Ensembles of Neural Networks", University of Regina, Regina, SK S4S 0A2, Canada, 2004.

[9] Rao H. Sudarsana, Ghorpade Vaishali G., Mukherjee A. , "A genetic algorithm based back propagation network for simulation of stressstrain response of ceramic-matrix-composites", Computers and

Structures, Pergamon Press, Inc, Jan. 2006 vol 84.
[10] Abraham Ajith et. al., “Soft Computing Models for Weather Forecasting” 2002
[11] Chun Lu, Bingxue Shi, “Hybrid back propagation/ GA for Multilayer Feedforward Neural Networks”, Inst. of Microelectronics, Tsinghua University Beijing 100084, China IEEE, 2000.
[12] Ileană Ioan, Corina Rotar, Arpad Incze, “The Optimization of Feedforward Neural Networks Structure using Genetic Algorithms”, Proc. 2004 Int. Conf. on Theory and Applicat. of Math. and Informatics, Thessaloniki, Greece, 2004.
[13] Sarangi1 Pradeepta et. al., “Short Term Load Forecasting using Artificial Neural Network: A Comparison with Genetic Algorithm Implementation”, ARPN J. of Eng. and App. Sci., Nov. 2009, vol. 9.