An Augmented Genetic Algorithm for Auto Corrective Prediction Model for Pest Attacks

Kumar Saurabh Bisht and Sanjay Chaudhary

Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar – 382007
E-mail : kumar_bisht, sanjay_chaudhary}@daiict.ac.in

ABSTRACT

Genetic algorithms or GAs are adaptive search algorithms. These algorithms are based upon the principles of evolution and natural selection. GAs are adept at searching large, non-linear search spaces and efficiently determining near optimal solutions in reasonable time frames by simulating biological evolution. We propose a model based on neural network for predicting pest attacks. This model takes a different approach where GA is used to prune faulty input training samples and also to retain and induce new good samples using reproduction operators: mutation and crossovers. The results show an improvement in the prediction error margin and convergence time in leaning phase of the model.

Keyword: *Feedforward Neural Network, Genetic Algorithm, Backpropogation algorithm*

1. INTRODUCTION

Pests are known as one of the main reasons to affect agriculture produce negatively. It is well known that pest diseases are cyclical phenomena as their development and attacks are in relation to host crop cycle and the environment. Several work [1][2] in past has been aimed to understand this pest dynamic and to establish correlations between various parameters pertaining to agricultural operations regarding crops, farming activities, weather and pest activity. Various approaches have been proposed to establish these correlations in order to address the major pertinent research problems of developing a feasible model for pest attack prediction.

This paper discusses a prediction model based on Neural Network and Genetic Algorithm. The model has a different two-prong approach for the subject problem. The error in the prediction is used to correct the model while the genetic algorithm based module prunes out the faulty training samples and inducts fresh training samples from database based on the fitness function. The fitness function determines the quality of the training sample.

2. NEURAL NETWORK

Neural Networks [3][5][6][7][8][10][11][13][16][17] or NNs are algorithms for optimization and learning, based on concepts of working of brain. A neural network is a highly parallelized dynamic system that consists of directed graphs. The topology is such that it generates the output by means of a reaction of its state on the given input data.

Generally a neural network consists of following component:

- A directed graph of set of nodes.
- A weight associated with each link/arcs interconnecting the nodes.
- A transfer function associated with each node. The transfer function can be expressed as:

$$f(\Sigma w*l, \Theta)$$

where "w" is associated with each link "l" connecting the node with other nodes and Θ is the activation function of the node. The nodes received inputs from previous layers and applied the weighted sum of these inputs to this activation function.

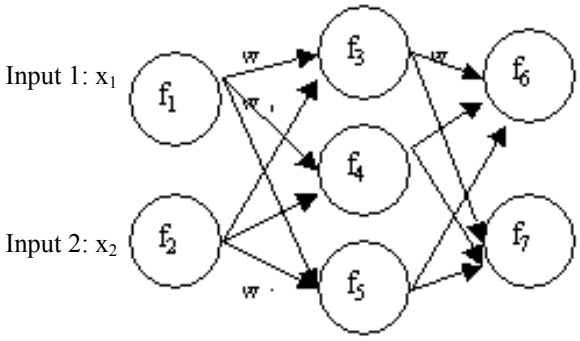

Fig. 1: A three layer Neural Network

Mathematically the output a node in neural network can be represented as a weighted sum of the activation functions. For example the output of f_3 node in second layer can be expressed as:

$f_{3\ output} = w_{13}\ f_1(x_1) + w_{23} f_2(x_2)$.

w_{13} = weight of link connecting nodes f_1, f_3.

w_{23} = weight of link connecting nodes f_2, f_3.

Similarly the output of other nodes can be calculated and perpetrated in the network till the output layer.

The layers of a neural network in general are of three kinds:

- Input Layer: This layer is the set of nodes, which receive data input. This layer can be used to convert data into required range, depending upon the activation functions of the network nodes.
- Output Layer: The nodes, which gives the final output, constitutes output layer. Again this layer can be used to change the output data to the required range depending upon the function applied at input layer.
- Hidden Layer: Apart from input and output all other nodes in different layers constitute hidden layer. These nodes do the processing of data.

Similar to human brain, a neural network has the ability to recognize and learn a pattern in a given data set. The learning of the network is encoded in the synaptic weights of the interconnected nodes. When a new input pattern different from the previously learned pattern is presented to the network, the network through changing of strengths of synaptic weights learns the new pattern.

An important part of the neural network is its training. The network structure undergoes learning process during which nodal weights are adjustment. During learning face, the network is presented with training samples of data repeatedly. The output of the network is compared with the actual output of training data set and the error is used to modify the network weights.

3. GENETIC ALGORITHM

Genetic algorithms [5][7][9][11][12][13] or GAs are adaptive search algorithms. GAs are based upon the principles of evolution and natural selection. They are adept at searching large, non-linear search spaces. Genetic algorithms search large and complex search spaces to efficiently determine near optimal solutions in reasonable time frames by simulating biological evolution.

In general, genetic algorithms consist of following components:

– Encoding of problem into chromosomes
– A fitness function that evaluates chromosome
– Operators to generate new population: mutation, cross-over

A genetic algorithm operates according to the following steps:

- Population initialization: sets of initial chromosomes are generated
- Evaluation: Each member of the population is evaluated
- Reproduction: Generation of new sets of chromosomes

When a genetic algorithm has appropriate encoded solutions of a problem and operators that can

generate better new chromosomes from parent chromosomes, the algorithm generates population of better and better chromosomes resulting in the convergence to global optimum. As genetic algorithm can improve the current best candidate monotonically they do not have the problem of local minima trap.

Local minima traps are a common problem associated with Neural Network. It happens when the learning phase of the network keeps oscillating between two error peaks and therefore fails to converge to optimal solution in learning phase.

So there can be a case where like in fig 2. the learning phase may oscillate between the marked local minima and will not be able to achieve optimum solution.

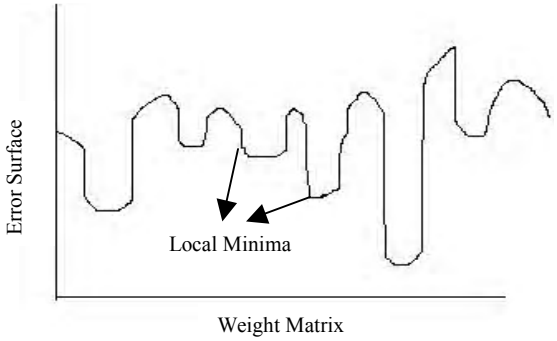

Fig. 2: An error surface with respect to weight matrices.

4. MODEL AND METHODOLOGY

A neural network design is statistical in nature therefore an appropriate tradeoff between complexity of the model and performance of the model is needed. In the context of backpropagation learning this tradeoff can be realized as [16]:

$$R(w) = \lambda e_c(w) + e_s(w) \qquad ...(1)$$

λ is a normalization parameter representing the relative importance of the complexity term (e_c) with respect to the performance-measure term (ε_s).

ε_c is the complexity factor, which depends upon the network model. For our network complexity we used heuristic approach and after repeated trial-error method decide the layer and node composition (details of model in 4.1.1).

$\varepsilon_s(w)$ is performance measure, in back-propagation model learning context. It is typically defined as a mean-square error of the output and forms the basis of network training. This factor depends on both network model and input data. One of the main reasons of error, in the prediction model is impurity of training data. A faulty training data can impair the learning of model and cause increase in prediction error.

The model we propose here uses a different approach by encoding the entire training database with GA. Earlier work [11][12] in the area of GA optimized neural learning is concentrated on improving of network learning, i.e. correction of weights, with GA. Our approach uses GA to prune and refine the training data set of the model. Genetic Algorithm is used to identify the faulty input samples i.e. the samples that causes high prediction error and based on the fitness function new samples are induced in the training sample set and faulty sets are pruned. The proposed model is based on a feed forward neural network. It consists of layers of nodes and has the topology such that there are no closed paths. We have used backpropogation algorithm [7][10] to train the network weights.

Levenberg-Marquardt (LM) [3] optimization is used for calculating cost function over the model parameters i.e. neural net weights and fitness function of genetic algorithm module. The cost function we use is based on optimized LM proposed by Wilamowski et.al [4].

$$f_{co}(x) = \sum_{j=1}^{n} [\Sigma (\Delta E)^2]^2 \qquad ...(2)$$

n = number of training samples.

ΔE is the error margin in the prediction of value of the model output:

$$\Delta E = o^j_{exp} - o^j_{train} \qquad ...(3)$$

o^j_{exp} = output predicted by model. Corresponding to j^{th} training sample.

o^j_{train} = output according to j^{th} training sample in the database.

4.1 Neural Network Layer

The feedforward neural network of the proposed model consists of five layers:

(1) Input Layer: This layer consists of four nodes.

(2) Hidden Layer: There are three hidden layers. First hidden layer has 6 nodes while second one has 12 and third has 6 nodes.

(3) Output Layer: The output layer has one node

The activation function of the 2 hidden nodes is:

$$f_h(x) = (1+\exp(-x))^{-1} \quad ...(4)$$

The activation of the input node is:

$$f_i(x) = x / I^i_{max} \quad ...(5)$$

I^i_{max} = highest value of i_{th} type input parameter in training sample.

The activation of the output node is:

$$f_o(x) = x / O^i_{max} \quad ...(6)$$

O^i_{max} = highest normalized value of output corresponding to of ith training sample, i.e.

$$O^i_{max} = \max(O^i). \quad ...(7)$$

$$O^i = O^i_{train} / \max(O^i_{train}). \quad ...(8)$$

$$i \in (1, 2, \ldots n)$$

n = number of training samples.

o^j_{train} =output corresponding to i^{th} training data tuple.

The input layer has no arcs to them and the output layer has no arcs away from them. Any path traversed from input to output node traverse the same number of arcs. N^{th} layer node connects to all nodes of $(N+1)^{th}$ layer nodes.

4.2 Parameters used in Neural Network

The backpropagation algorithm (details in 4.3.2) we used provides an approximation to the trajectory of the network weight vector space calculated by gradient descent method [8][10]. The learning of the model is controlled by various parameters, which defines model characters like learning rate and weight correction response. These parameters are:

- $\acute{\eta}$ = Learning rate
- δ = Local gradient
- Δw = Weight correction
- ψ = Momentum constant

Learning rate determines the size of the weight adjustment made at each training iteration and hence influences the rate of convergence.

Moment constant effects weight change process. It get the learning through one or local minima and get it into global minima.

These parameters are related as:

$$\Delta W_{ji}(n) = -\acute{\eta} * \delta_j(n) * y_i(n). \quad ...(9)$$

ΔW_{ji}=weight correction to the weight that connects the neuron i to the neuron j

$y_j(n)$ = input signal of neuron j i.e. output of neuron i.

δ_j = Local gradient

These parameters affect the learning of network as shown here:

$$\Delta W_{ji}(n) = \psi * \Delta W_{ji}(n-1) + \acute{\eta} * \delta_j(n) * y_i(n). \quad ...(10)$$

at n^{th} iteration

4.3 Augmented Genetic Algorithm

4.3.1 Genetic Algorithm Module Component

The genetic algorithm module of our model has following component:

- Encoding: Database is encoded in the form of chromosomes. An initial population of 'N' chromosomes is generated with each chromosome consisting of 'S' number of genes where 'S' is total number of data tuples in database. Each gene represents one tuple and has Boolean value of 'true' or 'false'. Only if the value of the gene is true, the tuple it represents is made available for training set.
- Reproduction Operators: There are two operators used mutation and crossovers. In mutation value of genes are set whereas in crossovers the cross between two chromosomes

produces new chromosome representing new training set.

- Pruning: In addition to reproduction operators above we introduce new operator that removes the gene from chromosomes and introduce new gene representing different tuple. The pruning is done based on the fitness of a chromosome. The fitness parameter of a chromosome is a function of χ and Δ: $\Gamma(\chi, \Delta)$.

χ = standard acceptable error of the model

Δ = error corresponding to a tuple represented by a gene

4.3.2 Gene fitness function

The fitness function of the genetic module determines the number of genes available for training in a chromosome and the associated error of the network with the chromosome. We have incorporated two parameters "trainingGain" and "inputGain" in the function. trainingGain controls the number of training cases that are discarded corresponding to an error in network. trainingGain determines the threshold error value for a gene to be set false. This parameter accounts for the fact that not every erroneous gene is impure tuple. The parameter is set for different experiments and error peaks in those experiments is used to fine-tune the parameter values subsequently. Input Gain controls the number of genes initially having a boolean value "true".

4.4 Algorithms

4.4.1 Prediction algorithm

```
Prediction
{
 t = number of tuple in database;
initialize P(x) ; //initial pool of population of chromosomes, 0< x <= t, P(x) is a chromosome
while (not termination condition: number of training cycle or convergence to acceptable error)
    {
I(n) // pool of chromosomes selected for training based on fitness; initially I(n) = P(x)
while (not termination condition : acceptable standard error)
    {
    input tuple to neural network from I(n);
    calculate output;
calculate error;
backpropogate algorithm(); //to modify weights of links;
fitness();  //fitness function to evaluate fitness of chromosome 'c',  c ε I(n);
remove();  //remove bad genes;
}
//generation of new population
    crossover();
    mutation();
for all x chromosome in the population P(x)
    {
        calFitness(); //calculate fitness;
    }
  }
}
```

- The fitness() function is based on LM optimization as discussed in eq (2)
- calFitness() calculate the fitness of a chromosome in a training sample based on number of bad genes i.e. gene denoted by 0 or "false" in chromosome string, error in the network and standard acceptable error.
- remove() function makes the representation of gene to "false" i.e. unavailable as training sample.
- insert() function makes the representation of gene to "true" i.e. available as training sample, used for mutation.
- crossover() and mutation() are standard genetic operators. crossover() makes chromosomes interchange gene value at different points along length thus certain genes are removed i.e. represented as "false" and certain genes as "true" while some genes get duplicated.

For ex. two chromosome strings

S1 = 001101 and

S2 = 101010 cross each other at positions 1, 4 and 6, so one of the possible new string S3 = 101000.

Similarly mutation is randomly inverting the representation of genes.

4.4.2 Backpropagation Algorithm

backpropogation algorithm

{

initialize all weights;

while(terminating condition)

{

for each training sample

{

for each hidden and output layer node j

{

$I_j = \Sigma w_{ij} * f_i(x)$(or $f_o(x)$) + Φ ;

//Calculation of node input of each unit in j layer receiving data from nodes in previous layer i.as discussed in Section 2.

and where Φ= bias of the node.

} *//end of for*

Error of the model = O(1-O)* ΔE (eq 2) ;

//where O = output of the model

for each hidden layer unit compute ;

//moving backwards that is from second hidden layer to first hidden layer.

updation of network weights;

updation of bias of nodes;

} *//end of for*

} *// End of outer while*

} *//End of back-propagation*

5. IMPLEMENTATION

The data to test the model is in MS xls format. The data is surveillance data set for the chickpea crop provided by ICRISAT, Hyderabad. A data tuple consists of following data:

- Weather Data: Min.(Tmin), Max.(Tmax) temperatures, Relative Humidity(H), Rainfall (R).
- Pest incidence: Larvae per plant (L)

To decrease the granularity of the data weekly mean of T_{min}, T_{max}, rainfall, humidity and larvae/plant is recorded. The pest surveillance data set for the chickpea crop was collected over a period of 11 years (1991-2001). Each tuple in the data set is of the form <T_{min}, T_{max}, Humidity (H), Rainfall (RF), Larvae/plant (L)>, where each value represents the weekly mean.

5.1 Data Processing

Since the range of activation function of nodes (eq. 3), lays from 0 to 1 the input data set is ranged to the required values before testing. The values are normalized to make the range appropriate. The input layer and output layer activation function does the normalization part. The network was trained with 300 tuples of data set iteratively with 6 different sets of 50 tuples. Thereafter the model was checked with another 50 set of tuples.

6. EXPERIMENTS AND RESULTS

The following criteria were the basis of the experiments conducted:

- Convergence of the learning phase
- Comparative performance between Augmented GA model and without GA model.
- Error in predicted value

Table 1: Convergence Iterations

Training Set Number	Augmented GA model iterations	Non GA model iterations
1	249	311
2	457	949
3	652	630
4	312	325
5	982	Learning phase did not converge
6	549	526
7	823	971
8	297	Learning phase did not converge
9	451	832
10	791	1041

(1) The table 1 shows the iterations required for convergence in learning phase of the Augmented GA and Non GA model.

The results show the Augmented GA model takes lesser number of iterations over a set of training sample to converge and learn. The result also shows that augmented GA model can avoid local minima trap more effectively.

(2) Comparative performance

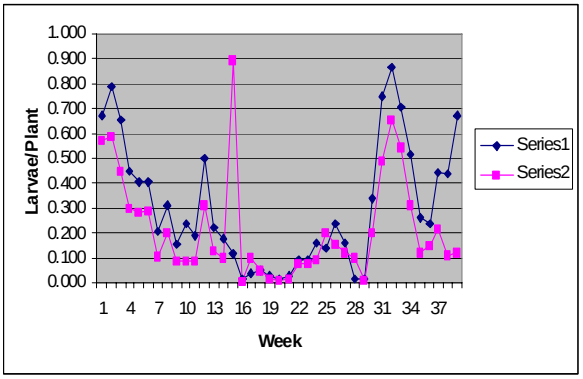

Fig. 3: Actual Value v/s Predicted Values by Non GA

The figure shows that the Non GA model successfully predict values with good approximation.

Series1 = Actual Values

Series2 = Predicted Values Non GA model

Peaks: Sudden change of pattern; Error

Tailing Effect: Error increases at the far end i.e as the period distance between the training samples and the prediction samples is increased the error margin increases.

Error Peaks: The sudden peaks in the predicted value show sudden change of pattern. The Non GA model falters around sudden change of pattern and prediction error is high

Series 1: Real Data

Series 2: Predicted Value by Augmented GA model

Series 3: Predicted Value by Non GA model

The result shows that the proposed model is able to predict more accurately than the Non GA model

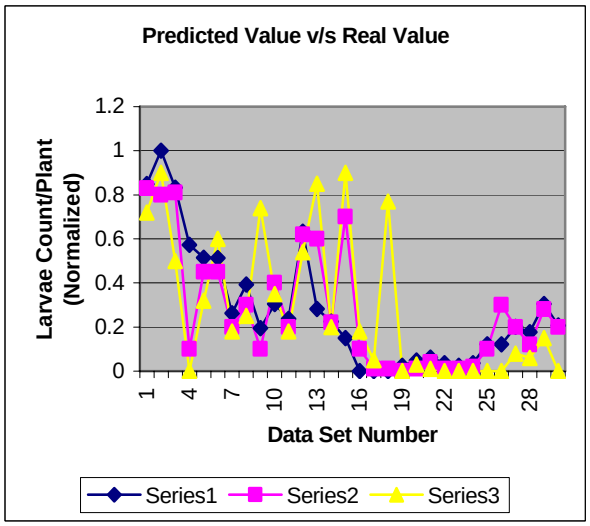

Fig. 4: Predicted values of both GA and Non GA model

(fig. 4). Also it is to note the peaks in the graphs where the prediction is better than the Non GA model, although not very correct with respect to real value (fig. 4) but the error margin is relatively low than the non-GA model.

It is to note that the performance of Augmented GA model around error peaks shown here is after fine-tuning of the parameter trainingGain. The parameter eventually decides the fitness of a chromosome and thus overall population. The fine-tuning of this parameter with repeated experiments ensures that probability of correct tuple being removed is high.

(3) Error in predicted value

Table 2: Error Performance

	Min. Error Percentage (approx.)	Max. Error Percentage (approx.)	Average Error Percentage (approx.)
Augmented GA Model	2	33	22
Non GA Model	12	41	30

Table 2 shows that the augmented GA model has higher accuracy of prediction in comparison to Non GA model with comparative lower error percentage in overall prediction.

7. CONCLUSION AND FUTURE WORK

The experiment result shows that the Augmented GA model is better than its non-GA counterpart in terms of prediction and learning traits. Also in the GA model the number of error peaks was less, in all the number of peaks for the 100 different testing, there was difference of 9 peaks in a best case i.e. 19 peaks in GA model in comparison to 28 peaks in the case of non-GA model. The tailing effect was similar in both the model and therefore on an average a prediction for an advance period of 7-10 days was in fair range of error margin.

Our current approach in establishing the structure for the network is based on heuristic approach. A better way to implement this modeling would be error dependent dynamic modeling, whereby the number of layers and nodes can be determined dynamically in the training phase. So depending upon the error, the number of nodes and layers in the network can be reconfigured repeatedly to achieve higher accuracy. Although this approach can increase the complexity of the overall model but a reasonable tradeoff between the complexity and real term prediction is where the future work can be concentrated.

8. ACKNOWLEDGMENT

We would like to thank Dr. V Balaji, Head, Knowledge Management and Sharing and Dr. G. V. Ranga Rao, Special Project Scientist (IPM), ICRISAT, Hyderabad, India for providing their domain expertise and cooperation in providing details about the problem, database for testing and related material

REFERENCES

1. Gupta Rajat and Narayna BVL, "Understanding Helicoverpa armigera Pest Population Dynamics related to Chickpea Crop Using Neural Networks", IEEE Computer Society Press, 2002.
2. D.K.Das, T.P.Trivedi and C.P.Srivastava. 2001. "Simple rules to predict attack of Helicoverpa armigera on crops growing in Andhra Pradesh", *Indian Journal of Agricultural Sciences* 71:421-423.
3. Simon Haykin, *Neural Networks: A Comprehensive Foundation*, Pearson Education, 2001.
4. Wilamowski Bogdan, Kaynak Okyay, Iplikci Serdar, Efe Onder M, "An Algorithm for Fast Convergence in Training Neural Networks", *IEEE, 2001*
5. Hand David, Mannila Heikki and Smyth Padhraic, "*Principles of Data Mining"*, MIT Press.
6. Dunham, Margaret H, *Data mining Introductory and Advance Topics*, PHI,
7. Kai Fu and Wenhua Xu, "Training Neural Network", *IEEE Computer Society Press, 1997*
8. Jiawei Han and Micheline Kamber, *Data Mining Concepts and Techniques*, Morgan Kaufmann, 2001.
9. Anthony Martin, *Discreet Mathematics of neural Netrwork*, SIAM,,2001
10. Kozma R, Kitamura M, "Anomaly Detection by Neural Network models*", IEEE Computer Society Press, 1992*
11. Rajshekhran S,Pai VijayLakshmi, *Neural Networks, Fuzzy Logic and Genetic Algorithm*, PHI,2003
12. Yang Lei, Dai Yu, Zhang Bin, Gao Yan, "A Genetic Algorithm Optimized New Structured Neural Network for Multistage Decision-Making Problem", *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05), IEEE Computer Society, 2005*
13. Yan Zhu, Jian Chen, "Studies on Genetic Multilayer Feedforward Neural Networks and the Development of GMNN", IEEE, 1999.
14. Fiszelew, A., Britos, P., Ochoa, A., Merlino, H., Fernández, E., García-Martínez, R., "Finding Optimal Neural Network Architecture Using Genetic Algorithms", *Advances in Computer Science and Engineering Research in Computing Science 27*, 2007
15. Saravanan,R, *Manufacturing Optimization through Intelligent Techniques*, Taylor & Francis, 2006
16. Galushkin I Alexander, *Neural Network Theory*, Springer, 2007
17. Chow S W Tommy, Cho Yeung Siu, *Neural Networks and Computing*, Imperial College Press, 2007.