AFLATOXIN PREDICTION USING A GA TRAINED NEURAL NETWORK

C.E. Henderson[a], W.D. Potter[a], R.W. McClendon[b], and G. Hoogenboom[b]
AI Center[a] and Bio & Ag Engineering Department[b]
University of Georgia, Athens GA 30602
dpotter@ai.uga.edu, rwmc@bae.uga.edu, gerrit@bae.uga.edu

ABSTRACT

Predicting the level of aflatoxin contamination in crops of peanuts is a task of significant importance. Backpropagation neural networks have been used in the past to model this problem, but use of the backpropagation algorithm for training introduces limitations and difficulties. Therefore, it is useful to explore alternative learning algorithms. Genetic algorithms provide an effective technique for searching large spaces, and have been used in the past to train neural networks. This paper describes the development of a genetic algorithm/neural network hybrid in which a genetic algorithm is used to find weight assignments for a neural network that predicts aflatoxin contamination levels in peanuts based on environmental data.

INTRODUCTION

The presence of aflatoxin contamination in peanuts has been connected with a number of health concerns, including carcinoma and necrosis of the liver (U.S. FDA, 1992). Efforts have been made to develop a method to predict the level of contamination expected for a crop (Cole, et al., 1985). It is believed that a relationship exists between environmental factors and aflatoxin levels. Specific factors that affect toxin levels include soil temperature and drought. For these reasons, there is interest in developing a tool that predicts aflatoxin levels based on environmental data. Such a tool could be used to guide crop management decisions and to select a harvest date that minimizes contamination levels.

The use of artificial neural networks (ANNs) is one way that predictive tools have been constructed for the aflatoxin problem, but the process of constructing neural network models can be difficult (Parmar, et al., 1997). Construction of neural networks requires setting values for the connection weights. Searching for connection weights, known as learning or network training, is a complex task. The backpropagation algorithm is an effective technique for network training but has a number of shortcomings, including high susceptibility to getting stuck in local minima.

Genetic algorithms are search methods based on the mechanics of natural selection and natural genetics (Goldberg, 1989). They are effective global search techniques that avoid many of the shortcomings of more traditional hill climbing searches and were chosen in this experiment to evolve weights for a neural network.

Genetic algorithms are a method for searching large, complex spaces, and have been used to train neural networks. By linking the global search capabilities of the genetic algorithm with the modeling power of artificial neural networks, a highly effective predictive tool for the aflatoxin problem can be constructed. This paper describes the use of a genetic algorithm/neural network hybrid that trains networks for the aflatoxin problem.

Fixed architecture, three layer feedforward networks with logistic activation functions were used to develop the models. As in a previous study by Parmar et al. (1997), inputs for the networks were soil temperature, drought duration, crop age, and accumulated heat units. The value for accumulated heat units was calculated based on a threshold temperature of 25°*c*.

There were two neural network models developed for the problem. One network (model A) was trained on all available data, while the other (model B) was trained using only data from undamaged peanuts. The network for model A produced r^2 values of 0.74, 0.83, and 0.21 for training, testing, and validation data sets. The model B network produced values of 0.45, 0.82, and 0.41. The implementation demonstrated that genetic algorithms can be used to train effective networks for the aflatoxin problem. The genetic search has the advantage that setting parameters is easier than for backpropagation, and the genetic trainer does not suffer from many backpropagation shortcomings, such as the tendency to stick in local minima.

A network design that has proven effective for aflatoxin prediction is the three-layer, feedforward neural network that uses backpropagation as the learning algorithm (Parmar et al., 1997). Such a network was chosen for this experiment.

The development of a model is started by gathering sets of patterns that are generally representative of the problem that is to be modeled. These data are allocated to a training set, testing set, and validation set. The training set can be used by backpropagation as a guide for making adjustments to the weights of the network. It is useful to employ the test set to decide when to stop training the network. The validation set serves to evaluate the performance of the network on new data after training is complete.

The process of developing a neural network model for a real world problem can be a time consuming, complex task. Many parameters must be set, and the effect that chosen values have on network performance is often poorly understood. The number of nodes in the hidden layer is one example of a problematic parameter. The problem of setting the values for the weights is handled using the backpropagation algorithm. However, using backpropagation introduces more parameters, including learning rate and momentum. It is known that learning rate and momentum can affect the quality of the final network, but there is no clear way to determine what values will prove to be best. These difficulties often necessitate a "try it and see" approach to neural network development, in which the investigator manually sets values for these parameters, observes how effective they prove to be, adjusts according to some heuristic, and tries again. This technique can be ineffective, because only a few parameter sets may be examined. Also, the investigator will rarely spend much time trying values that appear unpromising at first, even though these values could prove to be optimal ones. For these reasons, it is desirable to employ an automated search technique that avoids local minima while performing a global search of the parameter space. The genetic algorithm is appropriate for this problem and its use is described herein.

Genetic algorithms provide a mechanism that improves upon these limitations and have been used in the past as an alternative to backpropagation by Montana and Davis (1989), Whitley et al. (1990), Porto and Fogel (1990), Maniezzo (1994), and Fukumi et al. (1995). Also, genetic algorithms search globally and do not stick in local minima as readily as backpropagation. Genetic algorithms use only an objective function to guide the search and make no use of node activation functions. For this reason, genetic algorithms place no constraints on function selection. Genetic algorithms require fewer, simpler parameters to be set, so genetic algorithms provide an attractive approach to the problem of searching for network weight assignments.

Implementation

Both custom written code and modified versions of publicly available code were used for implementation of this project. Tools for the neural network portion were coded using Microsoft Visual C++. Features included the ability to allocate and de-allocate memory resources for networks dynamically and to set parameters such as number of hidden nodes, number of hidden layers, momentum, and learning rate at run time. Backpropagation was also implemented using Microsoft Visual C++ and included momentum and a learning rate parameter.

The implemented procedure for training via simple backpropagation proceeded as follows:

1. The network was trained on the training data set for a specified number epochs using backpropagation.
2. Mean absolute error was calculated for the network across all test set patterns.
3. If test set mean absolute error was the lowest ever, the network weights were saved.
4. If test set mean absolute error had not improved for a specified number of epochs, training was stopped and the weights that produced the best network throughout the training procedure were restored. Otherwise, go to step 1.

Genetic algorithm functions were obtained by converting and modifying a set of publicly available routines/libraries. GAlib, available from the Massachusetts Institute of Technology, was used. GAlib is available at http://lancet.mit.edu/ga/.

Versions of this experiment were developed to run under Windows 95 and UNIX. Results were gathered using a variety of hardware, including a Pentium PC and a number of Sun workstations.

Data Collection and Preparation

The following data collection descriptions are taken from Pinto (1996). The data used for this experiment were obtained from the United States Department of Agriculture Agricultural Research Service (USDA-ARS) National Peanut Research Laboratory (NPRL) at Dawson, Georgia. Measurements were taken from florunner peanuts that were grown in environmentally controlled stands. Following harvest, all peanuts were analyzed for aflatoxin contamination. Aflatoxin levels for entire stands were determined using the weighted average of the grade values.

Data sets were available for the years 1985 through 1995. Each observation consisted of the aflatoxin and environmental values for a specific plot and season. Environmental values included length of drought stress period (days), mean soil temperature (°*c*), crop age (days), and accumulated heat units (°*c* days). Drought stress was the number of consecutive days of drought conditions, while mean soil temperature was the mean temperature of the soil during this period. Crop age was the number of days from planting to harvesting. Accumulated heat units (AHU) was the accumulation of heat above 25°*c* during the drought period. This value was calculated by the following equation:

$$\text{AHU} = (\text{mean soil temperature} - 25) * \text{length of drought stress period}$$

This calculation was taken from Parmar et al. (1997) and was also used by Pinto (1996). These four environmental factors (drought duration, mean soil temperature, crop age, and accumulated heat units) are used as inputs for the neural networks developed in this project.

Because it was observed that the inclusion of damaged peanuts in the data introduced a great deal of noise, Pinto (1996) developed two neural network models. The first (model A) included data for both damaged and undamaged peanuts, while the second (model B) included only measurements for undamaged peanuts. The available data was used to produce two pools, one for model A and one for model B. For each of these pools, the data were sectioned into training, test, and validation sets. So that meaningful comparisons could be made between results, the current project used the same data sets as Henderson (1997), that were taken from Pinto (1996).

The purpose of this project was to compare the effectiveness of a genetic search for network parameters to traditional backpropagation. It is therefore necessary to have comparison results from models of the aflatoxin problem that did not include a genetic search. Henderson (1997) developed a common backpropagation neural network model using the same data as the current project. Therefore, this data will be used as a standard of comparison to evaluate the effectiveness of the GA/BPN approach.

Networks were evaluated by comparing predictions against target values for the patterns in the data sets and were developed separately for model A and model B data. R^2 values, the square root of mean squared error (RMSE), and the mean absolute error (MAE) were used as metrics. These values were calculated for the two models for each of the data sets.

Model Development

A fully connected, three layer, feedforward neural network using logistic activation functions was chosen as the modeling tool. The numbers of hidden nodes were set to constant values. For model A, the number of hidden nodes was set to 9. For model B, it was set to 8. These numbers corresponded to the best networks found in prior efforts. A genetic algorithm was designed to learn weight assignments for the neural network.

To employ the genetic algorithm to search for neural network parameters, there are a number of design issues that must be addressed. These issues are related to the representation, selection scheme, crossover scheme, mutation operator and the objective function. Choice of representation dictates many design decisions and should be examined first. The items to be represented are learning rate, momentum, and number of hidden nodes, which may be easily represented as real numbers. The network parameters to be evolved were therefore represented as strings of real values. There are a number of reasons to choose this representation over the traditional bit string for this problem:

1. The real valued representation has been used effectively for many similar problems.
2. Because the search space consists of sets of numeric values, each of which corresponds to a gene, decoding the chromosomes for evaluation is a simple task.
3. Crossover occurs only at real value boundaries, avoiding unwanted large leaps in value that are described below.
4. Genotypes are represented succinctly, consisting of three floating point numbers.
5. Because CPUs have built in floating point processors, manipulation of chromosomes is very efficient.

For a given genotype, the fitness of its phenotype is not unique for this problem. This is because the objective function employs backpropagation, which relies on an initial random seeding of the weight space to provide a search starting point. Hence, a genotype that can lead to an excellent network may receive a poor fitness rating due to unlucky initial weight assignments. The initial weights may be such that backpropagation gets stuck in a local minimum. To avoid this problem it is essential to evaluate promising genotypes multiple times. Luckily, the genetic algorithm provides just such a mechanism, selection. Many times selection is merely a way for a genetic algorithm to choose the most promising genotypes to use as parent chromosomes, but in this case there is an added bonus. If selection has no bias against selecting identical chromosomes from a pool, then the genotypes will get evaluated more than once. After a few generations, there will be many copies of the same fit chromosome in a population. Typically, this points to premature convergence and is to be avoided. However, we use an unusual mutator with a high mutation rate in this case. So even with a homogenous population, there is still strong pressure toward new diversity. For this reason, selection is of unusual importance to this implementation, as it provides a mechanism by which potentially strong genotypes may be evaluated more than once. Standard roulette wheel selection provides all of the features that are desirable and was chosen.

The objective function provides most of the evolutionary pressure for a genetic search, so great care was applied to its design. A genetic algorithm is limited in the information that it may exploit. In practice, this leads to many of its strengths. In contrast are such search techniques as backpropagation, which requires very specific resources (a differentiable activation function, for one) and may therefore only be applied to a limited number of problems. For this implementation, a given chromosome is decoded to produce a set of weight values, which are inserted into a network. The objective value returned is the square of the training set mean absolute error. Note that the genetic algorithm in this case strives to minimize the objective function.

Results

Networks were trained for model A and model B, and performance measures were calculated for training, testing, and validation data for each model. Hereafter, results generated by the genetic training algorithm will be called GA Trained values. GA Trained results were compared to the results from baseline experiments denoted Simple BPN and supply a baseline performance measure for simple backpropagation for the aflatoxin problem.

The GA Trained network performed better than Simple BPN for model A validation data and model B test data with respect to r^2. It performed about equally as well as Simple BPN for model A training data, model A test data, and model B validation data with respect to Mean Absolute Error. The only set for which the GA Trained network was clearly outperformed by Simple BPN was the model B training data set for r^2. So the results show that GA Trained networks perform competitively with respect to Simple BPN results for the aflatoxin prediction problem, see Figures 1-4.

It was mentioned previously that network performance on several data sets suggested noisy data. Our results show that the neural network made good predictions for all the test set data except for one marked exception. Because there are a small number of data points, the one highly inaccurate prediction had a strongly detrimental effect on the performance metrics for the set.

The strength of the GA Trained system is that it constructs networks in a manner that is free of many of the problems of backpropagation based approaches such as Simple BPN. Setting parameters for the GA Trained technique was significantly simpler than for backpropagation, and effective networks were found automatically. The only parameters that needed to be set were crossover rate, mutation rate, and pool size. This may not sound better than normal backpropagation, which also requires three parameter values (learning rate, momentum, and number of hidden nodes). However, the genetic search was fairly insensitive to these settings, finding good networks for every set of parameter values that was tried. Simple BPN, on the other hand, was highly sensitive to parameter settings and would find only extremely poor networks for the majority of values.

The results show the GA Trained network to be as effective as Simple BPN for training networks to predict aflatoxin contamination levels in peanuts. Development of the GA Trained network was also much easier than Simple BPN, requiring little human interaction to produce quality networks.

References

Cole, R. J., T. H. Sanders, R. A. Hill, and P. D. Blankenship. 1985. Mean geocarposphere temperatures that induce preharvest aflatoxin contamination of peanuts under drought stress. *Mycopathologia.* 91(1): 41-46.

Fukumi, M., S. Omatu, and Y. Nishikawa. 1995. Designing a neural network by a genetic algorithm with partial fitness. *Proceedings of the 1995 IEEE International Conference on Neural Networks.* 1834-1838.

Henderson, C. E. 1997. Using genetic algorithms to evolve neural networks. Master's Thesis. Artificial Intelligence Center. The University of Georgia.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization and machine learning.* Reading, MA: Addison Wesley.

Maniezzo, V. 1994. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks.* 5(1): 39-53.

Montana, D. J., and L. D. Davis. 1989. Training feedforward networks using genetic algorithms. *Proceedings of the International Joint Conference on Artificial Intelligence.* 762-767.

Parmar, R. S., R. W. McClendon, G. Hoogenboom, P. D. Blankenship, R. J. Cole, and J. W. Dorner. 1997. Estimation of aflatoxin contamination in preharvest peanuts using neural networks. *Transactions of the ASAE.* 40(3): 809-813.

Pinto, C. S. 1996. Prediction of aflatoxin contamination in peanuts using artificial neural networks. Master's Thesis. Computer Science Department. The University of Georgia.

Porto, V. W., and D. B. Fogel. 1990. Neural network techniques for navigation of AUVs. *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology.* 137-141.

U. S. Food and Drug Administration Center for Food Safety and Applied Nutrition. 1992. *Foodborne Pathogenic Microorganisms and Natural Toxins.* Washington, D. C.: U. S. Food and drug Administration.

Whitley, D., T. Starkweather, and C. Bogart. 1990. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing.* 14:347-361.

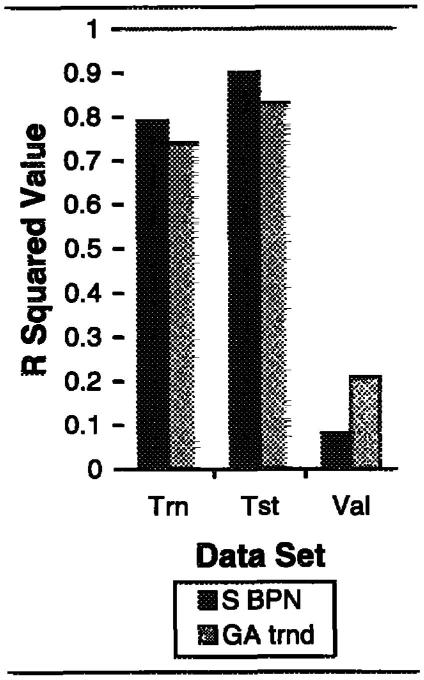

Figure 1. Best R squared values on data sets for model A.

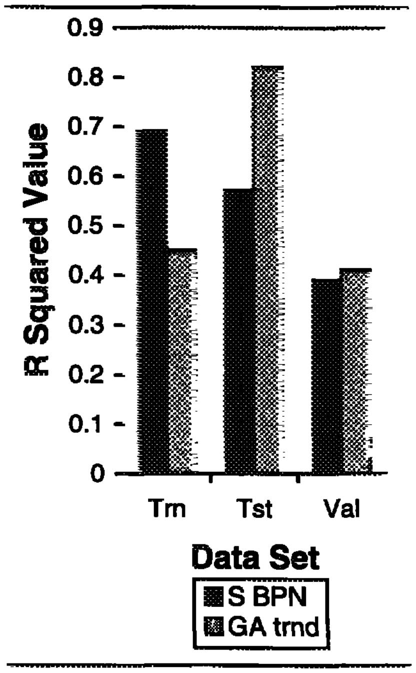

Figure 3. Best R squared values on data sets for model B.

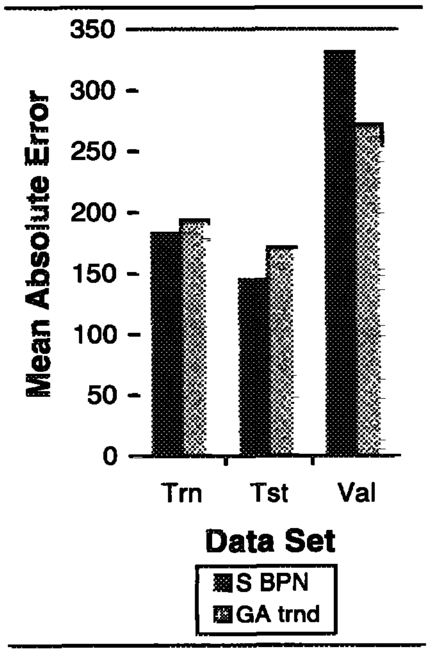

Figure 2. Best mean absolute error on data sets for model A.

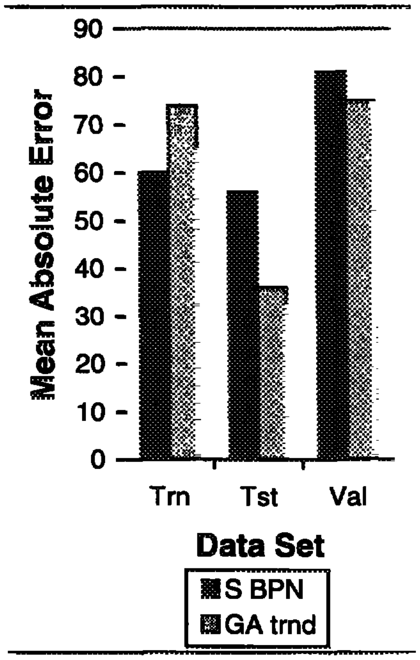

Figure 4. Best mean absolute error on data sets for model B.