Performance of a Neural Network: Mapping Forests Using GIS and Remotely Sensed Data

A.K. Skidmore, B.J. Turner, W. Brinkhof, and E. Knowles

Abstract

Neural networks have been proposed to classify remotely sensed and ancillary GIS data. In this paper, the backpropagation algorithm is critically evaluated, using as an example, the mapping of a eucalypt forest on the far south coast of New South Wales, Australia. A GIS database was combined with Landsat thematic mapper data, and 190 plots were field sampled in order to train the neural network model and to evaluate the resulting classifications. The results show that the neural network did not accurately classify GIS and remotely sensed data at the forest type level (Anderson Level III), though conventional classifiers also perform poorly with this type of problem. Previous studies using neural networks have classified more general (e.g., Anderson Level I, II) land-cover types at a higher accuracy than those obtained here, but mapped land cover into more general themes. Given the poor classification results and the difficulties associated with the setting up of suitable parameters for the neural-network (backpropagation) algorithm, it is concluded that the neural-network approach does not offer significant advantages over conventional classification schemes for mapping eucalypt forests from Landsat TM and ancillary GIS data at the Anderson Level III forest type level.

Introduction

In this paper, a neural network (specifically, the backpropagation algorithm) maps eucalypt forest vegetation. Neural-network models have previously been used with remotely sensed and other ancillary data, but the work frequently lacks details, and the results are mostly cited for Anderson Level I or II classifications (Anderson *et al.*, 1976). Anderson Level I refers to general thematic classes such as forest, water, or soil, while Anderson Level II subdivides these classes into sub-groups such as deciduous or coniferous forest. Repeating the subdivision process to Anderson Level III defines forest types. The success (correctness) of a classification needs to be considered in relation to the scale at which the thematic classes are defined. It is relatively easy to obtain an accurate map at Anderson Level I using standard classifiers, but difficult at Anderson Level III (Skidmore and Turner, 1988; Skidmore, 1989).

A study by Hepner *et al.* (1989) concluded that neural networks (NN) could map general land-cover types (such as water, land, forest, and urban at Anderson Level I) with greater accuracy than a conventional maximum-likelihood classifier when using Landsat Thematic Mapper (TM) data. Hepner *et al.* (1989) also used a textural measure in their classification scheme, which has been replicated in this study. When Fitzgerald and Lees (1992) repeated the approach of Hepner *et al.* (1989) in an Australian context, they found the neural-network algorithm also performed better when mapping general land-cover classes. Parikh *et al.* (1991) used Landsat TM imagery to map linear geological features. The neural network was trained using digitized lineament maps and was found to be superior to linear discriminant functions and k-nearest neighbors for this purpose. Civco (1993) mapped land covers from Landsat TM data at Anderson Level II and concluded that the neural-network approach was comparable to maximum likelihood. Omatu and Yosida (1991) mapped general classes (Anderson Level I) such as sunlit and shadowed forest, urban, water, and grass using a neural network (backpropagation algorithm), and reported good correlation between the areas correctly mapped by the neural network and the true area. The accuracy of the neural-network classifications was lower than that of the maximum-likelihood classifications.

The main objective of this study was to understand the behavior of neural networks (specifically the backpropagation algorithm) with remotely sensed and GIS data. In so doing, the usefulness of neural networks for classifying remotely sensed and GIS data was critically evaluated. A second objective was to map complex native forest at Anderson Level II and III using the backpropagation algorithm. From this work, it is hoped that others who may wish to also classify GIS and remote sensing data using neural networks may be able to find guidance in setting the various network parameters, and thus save time in developing heuristics for this purpose.

Study Area

The study area, located in the southeast forests of New South Wales, is approximately 20 km northwest of the Eden township. Topographic relief is moderate (Bridges, 1983). Precipitation is approximately 1000 mm per year (Keith and Sanders, 1990), and temperatures are mild year round with an average annual temperature of 15°C. The parent material consists of rhyolite and basalt outcrops (Ferguson *et al.*, 1979), as well as Ordovician metamorphic material. Soils are generally acid, highly weathered, and of poor to moderate fertility.

Vegetation of the Nullica study area is primarily dry and damp sclerophyll eucalypt forest, with the dominant species being silvertop ash (*Eucalypt sieberi*) and various stringybark species (such as *E. agglomerata*). The area is largely undisturbed, except for some low-intensity selective logging and, latterly, construction of some forest access roads.

A.K. Skidmore is with ITC, P.O. Box 6, 7500 AA Enschede, The Netherlands.

B.J. Turner is with the Department of Forestry, ANU, P.O. Box 1, Canberra, ACT 2601, Australia.

W. Brinkhof and E. Knowles are with the School of Geography, University of New South Wales, Sidney 2052, Australia.

Photogrammetric Engineering & Remote Sensing, Vol. 63, No. 5, May 1997, pp. 501–514.

0099-1112/97/6305-501$3.00/0

Description of the Neural-Network Algorithm

A backpropagation algorithm was implemented for a three-layer network (see Figure 1) consisting of an input, hidden, and output layer because

- most comparable studies used the backpropagation algorithm, or a derivative of the backpropagation, so its use allows a comparison with these results; and
- discussions with experienced colleagues revealed a consensus that the backpropagation algorithm is generally applicable and has good modeling capabilities.

Training data, consisting of the values for a grid cell (pixel), are presented to the neural network, together with a known land-cover class. The arrangement is similar to that of conventional supervised classifiers (e.g., maximum likelihood). For example, a training area of 15 pixels over a lake may be delineated from Landsat TM data; each pixel will have three brightness (DN) values associated with it and the output class is water. In this implementation of the backpropagation algorithm, each output class is assigned to an output node. For example, if five output classes were to be classified, Class 1 would be labeled as {1 0 0 0 0}, Class 2 as {0 1 0 0 0}, Class 3 as {0 0 1 0 0}, and so on. Each output node has an associated "target" value. In other words, a water class may be assigned to output node number 3, and be given a target value of, for example, 0.90; output nodes 1, 2, 4, and 5 would be set to a target value of 0.10. The water class would then be labeled {0.10 0.10 0.90 0.10 0.10}. Similarly, a forest class may be assigned to output node 2 with a target value of 0.90, while the other output nodes would have a target value of 0.10, that is, {0.10 0.90 0.10 0.10 0.10}.

The backpropagation algorithm comprises a forward and a backward phase through the neural-network structure. The first phase is forward, during which the values of the output nodes are calculated based on the GIS and remotely sensed data values input to the neural network. In the second phase, the calculated output node values are compared with the target (i.e., known) values. The difference between the value calculated for the node and the value of the target node is treated as the error; this error is used to modify the weights of the connections in the previous layer. This represents one epoch of the backpropagation algorithm. In an iterative process, the output node values are again calculated, and the error is then propagated backwards. The total error in the system is calculated as the root-mean-square error between the calculated value and the target value for each node. The algorithm continues until the total error in the system decreases to a pre-specified level, or the rate of decrease in the total system error becomes asymptotic.

A brief description of the backpropagation algorithm now follows; other useful references are works by Rumelhart and McClelland (1986), Pao (1989), Aleksander and Morton (1990), Kosko (1992), and Haykin (1994). The feed-forward stage starts with the remotely sensed and/or GIS input data (o_i) being presented to a node and multiplied by a weight (w_{ji}). The products are summed at the hidden nodes to produce a value z_j for the jth layer: i.e.,

$$z_j = \sum_i w_{ji} * o_i. \tag{1}$$

For a three-layer neural network, with the three layers lettered as i,j,k, and k being the output, z_k may be similarly calculated as for Equation 1.

In an attempt to mimic the output from a biological cell, the value of z_j is passed through a transfer function, which is often sigmoidal (Equation 2). The output from this activation function is

$$o_j = \frac{1}{1 + e^{-(z_j + \theta)/\theta_0}} \tag{2}$$

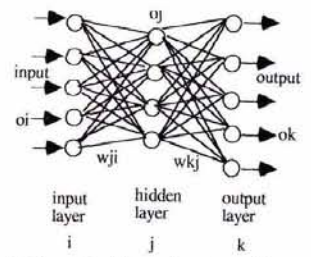


Figure 1. Neural network structure for a one-layer network.

where z_j is defined in Equation 1, θ is a threshold (or bias), and θ_0 is a constant. This adds non-linearity to network calculations, which is an important mathematical property allowing the network to solve some complex problems more accurately than linear techniques. A sigmoidal activation function for z_j is shown in Figure 2, for $\theta = 0$.

The calculation utilizing the sigmoidal function is repeated for each hidden node, and finally terminates after the o_k value is calculated for each output node. This represents the end of the feed-forward phase of the first epoch.

The backpropagation phase now commences. The basis for this process is that the initial output values almost never equal the target or desired value of the output node. The system error (E) is therefore calculated, which is the difference between the target value (t_{jk}) (or desired value as defined by the training area pairs) and the output value (o_{jk}). Note in Equation 3 that E has been aggregated into a single adjustment, where P is the number of training patterns: i.e.,

$$E = \frac{1}{2P} \sum_j \sum_k (t_{jk} - o_{jk})^2. \tag{3}$$

The aim is to reduce E by "back-propagating" the error from the output nodes to the hidden nodes, and from the hidden nodes to the input nodes. Backpropagation of the error is achieved by changing the weight of each node (in a backwards direction) using the following relationship:

$$\Delta w_{ji} = \eta \delta_j o_i \tag{4}$$

$$\Delta w_{kj} = \eta \delta_k o_j \tag{5}$$

for the jth and kth layer where

$$\delta_j = \frac{\delta E}{\delta o_j} \tag{6}$$

$$= o_j(1 - o_j) \sum_k (\delta_k \omega_{kj}) \tag{7}$$

and

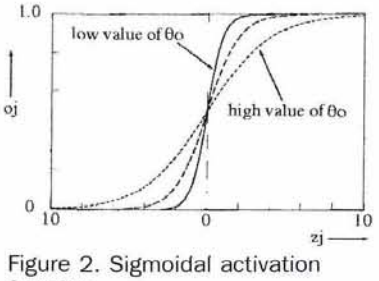


Figure 2. Sigmoidal activation function.

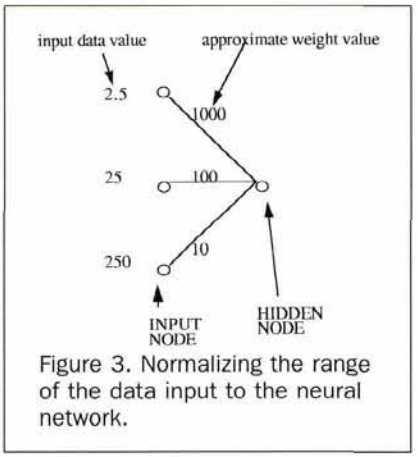


Figure 3. Normalizing the range of the data input to the neural network.

$$\delta_k = (t_k - o_k)(1 - o_k)o_k \quad (8)$$

where η is the learning rate constant.

A momentum term (α) may be added to increase the learning rate: i.e.,

$$\Delta\omega_{ji}(n + 1) = \eta\delta_j o_i + \alpha\Delta_{ji}(n) \quad (9)$$

where n is the presentation number (epoch).

The output from each node is calculated by the sigmoidal activation function (Equation 2). Over many epochs (or iterations), the total system error is reduced by this two-phase process of feed-forward, followed by error backpropagation.

Methods

Building the Data Set

The data set used comprises 190 field plots, a small number for training a neural network, but for a natural resource inventory, the density of plots is actually high. The field plots were sampled in a stratified random manner, strata being based upon geological types and topographic position. Plots were 0.10 ha in size, and at each site eucalypt species were identified, and heights were estimated by measuring the height of five trees per plot and visually estimating the remainder. Using the species data for each site, forest type classes were defined using the classification system developed by Baur (1965).

Natural eucalypt forests are a complex mix of species; the species form into natural groups (i.e., forest types). Traditionally, the species mix is "interpreted" from casual observation or aerial photographs, but this may be a major source of error. In contrast to earlier studies, we wished to control the potential error of traditional classification and interpretation methods. Therefore, in this study the species mix was verified using counts of each species in a plot.

A spatial database consisting of elevation, slope, aspect, topographic position, geology, rainfall, temperature, and remotely sensed data was geometrically corrected to a UTM projection, and interpolated to a 30-m grid. From these data, forest types were predicted, i.e., scrub, dry schlerophyll, damp schlerophyll, wet schlerophyll, and rainforest.

Normalizing the Input Patterns

The process of developing a neural-network application starts with selecting and modifying the input data in order to allow the neural network to reach a feasible solution in a reasonable time. Theoretically, the input data should be normalized to the same range, in order to speed convergence to a minimum error point in the network. This can be visualized in Figure 3, where the input data have different magnitudes. In order for each node to have the same order of magnitude effect on the output of the hidden node, the weights need to be inversely proportional to the input data values.

The problem was solved by normalizing the input data to a range between 0 and 1 using a linear contrast stretch (Richards, 1986).

Results

Introduction

In the following experiments, one system parameter was varied while holding other parameters constant, in order to highlight the effect of the varied system parameter on the performance of the neural network. The parameter settings for the different experiments are listed in Table 1. Note that n/a means not applicable, and is included in the table to indicate that a variable is indirectly varied by manipulating another variable. For example, for the "GIS vs TM only" experiment, the number of input nodes was varied, and that indirectly changes the number of hidden nodes in the network (remember that the hidden nodes connect to the input nodes).

Accuracy was measured for all experiments by randomly splitting the available 190 plots into a training data set and a testing data set. As shown in Table 1, the usual number of points used for training the network is 150, leaving 40 points to test the accuracy of the prediction. Accuracy is reported as the percentage of points correctly trained (i.e., training accuracy) or predicted (i.e., test accuracy) by the network.

Note that some of the variation in the percentage of correct training (and test) data stems from the method used to generate results from the neural network (e.g., Figures 8 and

TABLE 1. SETTINGS FOR CRITICAL SYSTEM PARAMETERS FOR THE EXPERIMENTAL TRIALS. NOTE THAT n/a MEANS NOT APPLICABLE.

Experiment	number of inputs	number of outputs	number of layers	number of hidden nodes	learning rate	momentum	number learning patterns
raw data	13	5	3	180	0.2	0.4	150
GIS vs TM only	V	5	3	n/a	0.2	0.2	150
random inputs	13	5	3	180	0.2	0.2	150
texture	V	5	3	n/a	0.2	0.2	150
# learn patterns	13	5	3	180	0.1	0.4	V
# hidden nodes and layers	13	5	3-5	V	0.8	0.2	168
# output nodes	13	V	3	n/a	0.8	0.2	150
# epochs	13	5	3	180	0.2	0.2	150
system error	13	5	3	180	0.2	0.2	150
learning rate	13	5	3	180	V	0.3	150
momentum	13	5	3	180	0.2	V	150
target value	13	5	3	180	0.2	0.5	150

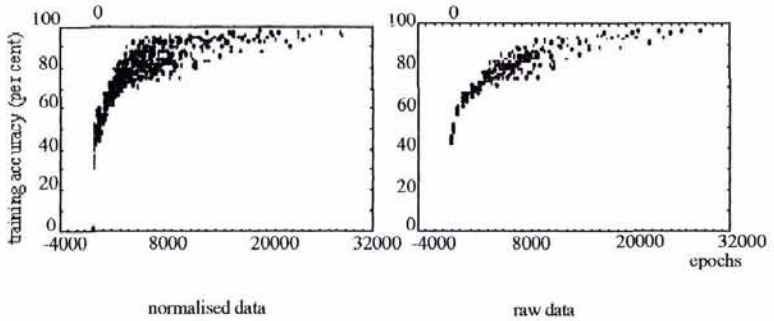


Figure 4. Relationship between number of epochs and the percentage of training data correctly predicted for the raw and normalized data.

9). The network was stopped each time the total system error dropped by 10 percent, and the critical system parameters (e.g., number of epochs, total system error, percentage of test data correctly predicted) were written to a file. The rationale was to explore how the neural network performed during the learning process, particularly for apparent outliers; for example, in experiments where the system produced a low total system error after a few epochs. Note that some experiments may have an initial accuracy of 0 percent (for example, see Figures 8 and 9). In these situations, the system error, by chance, started below the initial threshold set for the network to stop and report critical values.

To generate the following results, many experiments were executed using a network of 24 Sun workstations, a Silicon Graphics Power Challenge with four processors, and a (32-processor) Thinking Machines CM5 computer. The backpropagation algorithm was programmed in the C language so that the code could be easily integrated into future research projects. To check this algorithm, two public domain backpropagation algorithms were obtained; these generated the same results.

Use of Normalized versus Raw Data

A plot of the training data accuracy obtained using the raw data and the normalized data (Figure 4) indicates that normalized data reduced the number of epochs and, hence, the computational expense of processing. A research hypothesis that more epochs are required to obtain a training accuracy of greater than 90 percent when using raw data was tested with the Mann-Whitney U test. Stated formally, the null hypothesis is Ho: $\eta_1 = \eta_2$ versus the alternate hypothesis Ha: $\eta_1 > \eta_2$, where η_1 is the median number of epochs with a training accuracy of greater than 90 percent for the raw data, and η_2 is the median number of epochs with a training accuracy of greater than 90 percent for the normalized data. The null hypothesis was rejected at $p < 0.0001$, so we conclude that normalized data require fewer epochs to approach a high training data accuracy, compared with the raw data.

By normalizing the data, fewer epochs are required to obtain a small system error as the weights of the nodes have approximately the same range. Wilson (1991) commented that modifying the ranges of the input data caused the network to learn at different rates, and surmised that different ranges sometimes worked better because they utilize a larger percentage of the sigmoid function.

Randomized Input Data

If the training data are presented to a neural network in an iterative sequential manner, then the network may need to learn the spectral (and other) patterns of the training data, as well as the order in which the data were introduced. Does randomly presenting the input data to the network improve its performance? To test this research question, the order of data presented to the neural network was randomly varied (e.g., ABC, BCA, CBA) as well as input sequentially. The effect on the training and test accuracy of the random versus the sequential input are shown in Figure 5.

The exploratory data analysis (Figure 5) indicated little difference in training and test accuracy as a result of using random or sequential input data, though the test accuracy appears somewhat higher for the sequentially input data. A research hypothesis that random presentation of training data increases training accuracy was tested using the Mann-Whitney U test. Stated formally, the null hypothesis is Ho: $\eta_1 = \eta_2$ versus the alternate hypothesis Ha: $\eta_1 > \eta_2$, where η_1 is the median training accuracy for the random data, and η_2 is the median training accuracy for the sequential data. The null hypothesis was not rejected at $p = 0.05$, so we conclude that there is no difference in training accuracy for randomly versus sequentially presented input data. The Mann-Whitney U test was repeated for the test accuracy. Stated formally, the null hypothesis is Ho: $\eta_1 = \eta_2$ versus the alternate hypothesis Ha: $\eta_1 > \eta_2$, where η_1 is the median test accuracy for the random data, and η_2 is the median test accuracy for the sequential data. In contrast to the training accuracy, the null alternative was rejected at $p < 0.0001$, confirming that the test accuracy was higher for the sequentially presented data compared with the randomly presented data.

One explanation for the behavior of the neural network is that the input data used are large as well as complex, and the order of input/output pairs for the sequential experiment was well shuffled. Therefore, the network was not learning the order of the data presented to it.

Input of TM and GIS Data versus TM Data Alone

When only TM data were input to the neural network, the training accuracy appeared lower compared with using the combined TM and GIS data set[1] (Figure 6). A research hypothesis that the training accuracy was higher for all data compared with the TM data may be stated formally as Ho: $\eta_1 = \eta_2$ versus the alternate hypothesis Ha: $\eta_1 < \eta_2$, where η_1 is the median training accuracy for all data and η_2 is the median training accuracy for the TM data. The null hypothesis was tested using the Mann-Whitney U test, and rejected at $p < 0.001$; we conclude that the training accuracy is lower for the TM data compared with the use of all data. Interestingly, the test accuracy appeared higher for the TM data (Figure 6). This observation was also formally tested using the Mann-

[1]Note that "all data" comprised the TM and GIS layers (elevation, slope, aspect, topographic position, geology, rainfall, and temperature).

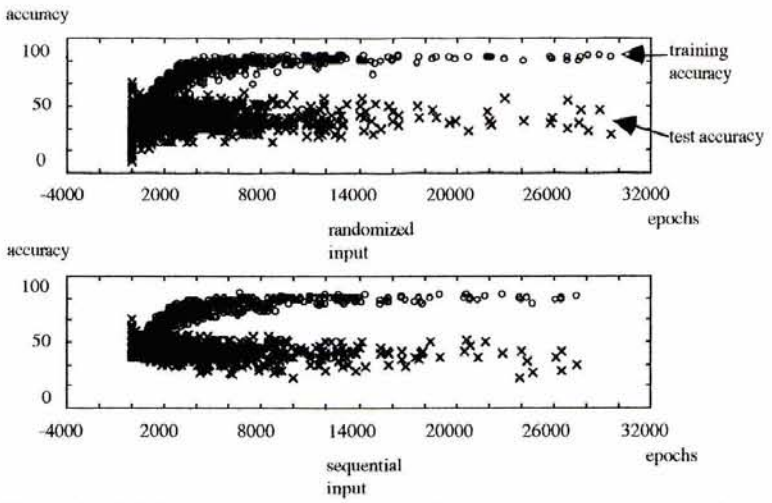


Figure 5. Training and test accuracy resulting from randomized input versus sequential input of the training data.

Whitney U test, which confirmed that the test accuracy is significantly higher for the TM data (compared with the combined TM and GIS data set) at $p < 0.001$. An explanation for this apparent contradiction may be made by analogy to the fitting of a high order polynomial using a few sample points; the training accuracy (fit) may be high, but the accuracy with unknown data (test accuracy) is reduced (Richards, 1986). As the TM data sample is smaller than the combined GIS and TM data set, a similar phenomena may be occurring here.

Another interesting observation from Figure 6 is that there are fewer plotted points for the TM data compared with the combined TM and GIS data set. This is because the neural network program stopped each time the total system error decreased by 10 percent. The TM data set rapidly converged to the minimum system error, because the data set was simpler and, hence, easier to learn compared with the combined TM and GIS data set.

Texture

Hepner *et al.* (1989) used texture as an input layer to a neural network. In the following experiments, two measures of texture are used: skew and variance (skew is the deviation of the distribution from symmetry, and variance is a measure of the spread of the data, within a moving window). The window size (across which texture is evaluated) was varied between 3 by 3, 5 by 5, and 7 by 7; in contrast, Hepner *et al.* (1989) used a 3 by 3 window.

An initial research question was whether the size of the moving window influenced the training and test accuracy of the neural network. Skew, as evaluated within the 5 by 5 moving window, had a significantly higher training and test accuracy than within the 3 by 3 and 7 by 7 moving windows for $p < 0.01$, as calculated by the Mann-Whitney U-test. In contrast, variance within a 7 by 7 moving window produced the highest training and test accuracy.

A combined data set of skew (evaluated within a 5 by 5 moving window), TM, and all GIS layers allowed the neural network to train faster compared with using only TM and GIS data (Figure 7). An asymptote on the training accuracy curve was achieved after about 1500 epochs for the combined skew, TM, and GIS data set, compared with approximately 6000 epochs for the TM and GIS data set.

A research hypothesis that the training accuracy was higher for the combined skew data set (i.e., skew, TM, and all GIS layers) compared with the GIS and TM data set may be stated formally as Ho: $\eta_1 = \eta_2$ versus the alternate hypothesis Ha: $\eta_1 > \eta_2$, where η_1 is the median training accuracy for the combined data set containing skew, and η_2 is the median training accuracy for the GIS and TM data. The null hypothesis was tested using the Mann-Whitney U test, and rejected at $p < 0.01$; we conclude that the training accuracy is higher for the combined skew data set compared with the GIS and TM data. Similarly, the test accuracy was significantly higher when combined skew was included (MannWhitney U-test at $p < 0.01$).

Figure 7 also indicates that training and test accuracies appear higher when variance was combined with the TM and all GIS layers. This was confirmed using the Mann-Whitney U test; we conclude that the training and test accuracies are significantly higher for the "variance" data set (i.e., variance combined with the TM and all GIS layers) compared with the GIS and TM data, at $p < 0.01$. Interestingly, the test accuracy for the (combined) skew data set was significantly higher than

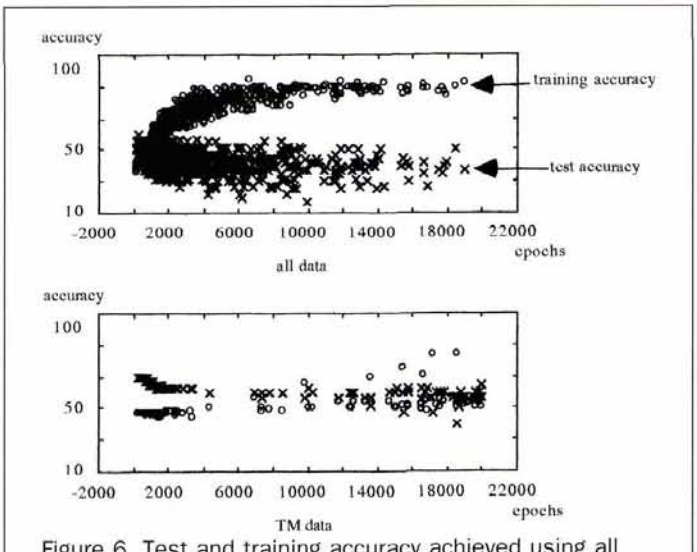


Figure 6. Test and training accuracy achieved using all data and the TM data alone.

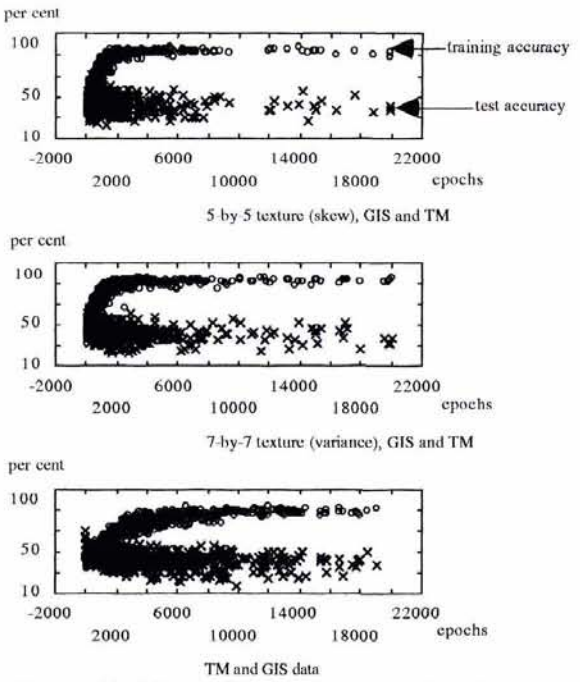


Figure 7. Training and test accuracies when texture (skew and variance) were added as input data layers.

for the (combined) variance data set (Mann Whitney U test at $p < 0.01$), but there was no difference in the training accuracy.

In conclusion, analysts should consider including texture (skew and variance) in classifications; for the data set used here, there was a statistically significant increase in training and test accuracy. Texture appears to provide additional information to discriminate classes.

Number of Learning Patterns

There is little variation in test accuracy with fewer learning patterns (i.e., number of test plots) (Figure 8). Variation in accuracy appears to be an artifact of the data set used; when the data set was modified by changing the order in which the learning patterns were presented to the neural network, maximum accuracy occurred at 90 learning patterns.

Similarly, there appears to be little variation in the accuracy of training data as the number of learning patterns increases (Figure 9).

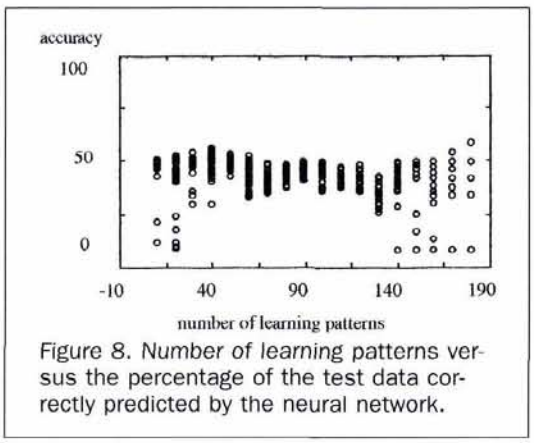


Figure 8. Number of learning patterns versus the percentage of the test data correctly predicted by the neural network.

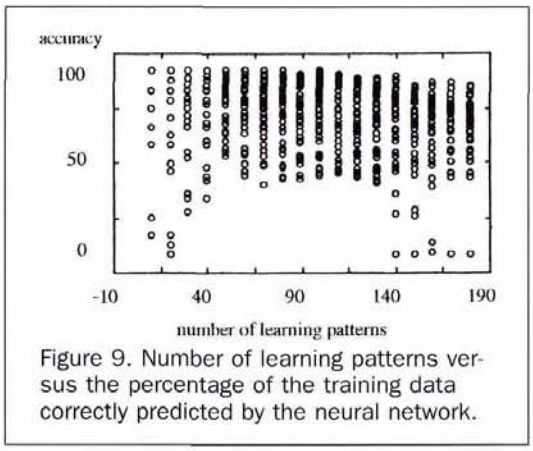


Figure 9. Number of learning patterns versus the percentage of the training data correctly predicted by the neural network.

Network Size

Number of Layers and Number of Nodes per Layer

The number of layers, as well as the number of hidden nodes per layer, affects the performance of a neural network. Figure 10 shows how the average training and test accuracy varies as the number of hidden layers increases from one to three, and the number of hidden nodes per layer rises from one to 50. It appears that higher training accuracies are obtained with three hidden layers, compared with two or one hidden layers. Also, as more hidden nodes are added, the training accuracy increases. In contrast to the training data, the average test accuracy declines as the number of hidden nodes increases. Test accuracy is lowest with one hidden layer.

Note for Figure 10, in order to smooth short-range fluctuations, the average training accuracy value was calculated for increments of ten nodes per layer; that is, the average accuracy was calculated for the range of one to ten nodes, 11–20, 21–30, 31–40, and 41–50 nodes per layer.

The relationships in Figure 10 were confirmed by a two-way analysis of variance. The research hypothesis was whether a significant difference in accuracy occurred (for both the training and test data) as the number of hidden layers and nodes per layer varied. In other words, it was of in-

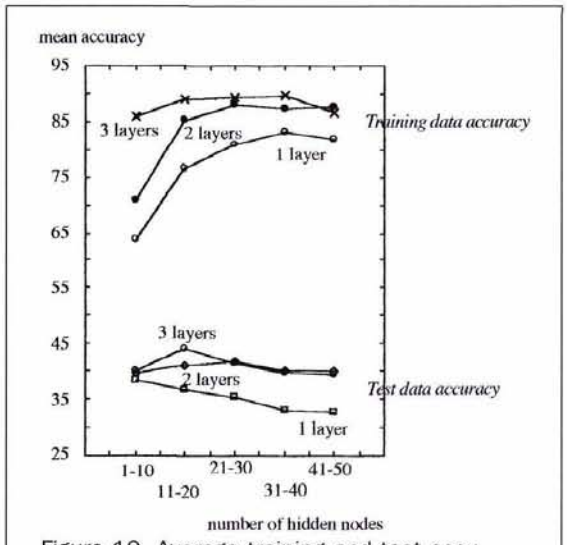


Figure 10. Average training and test accuracy, stratified by number of hidden layers and number of hidden nodes per layer.

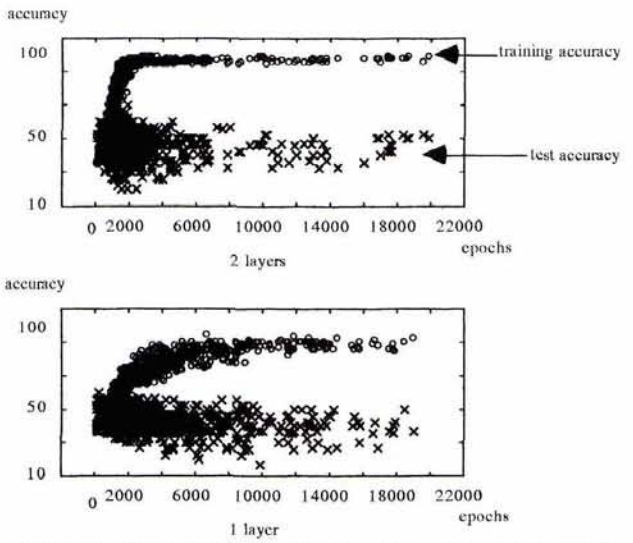


Figure 11. Number of epochs required to reach maximum training accuracy.

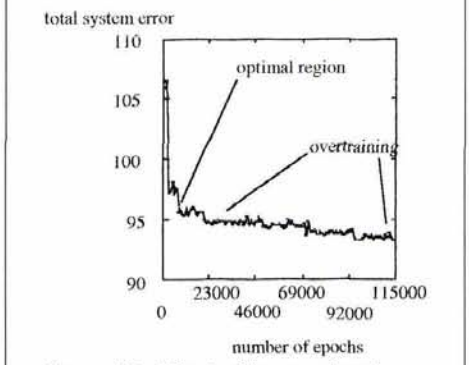


Figure 12. Effect of increasing the number of epochs on total system error.

terest to learn how the number of layers and number of nodes per layer (as well as any interaction effect) affects average training (and test) accuracy.

The first stage of the analysis checked that the assumptions of the test were not violated, specifically, that the dependent variables were normally distributed, and that variances were homogeneous within each group. These assumptions were tested by viewing histograms, and using a series of tests including Cochran C, Hartley, and Levene's test; all tests showed that there were no significant violations of the assumptions. A potentially more serious problem is correlation between means and standard deviations, because an extreme cell (i.e., mean value) may be present in the analysis of variance design which also has greater than average variability. The correlation between means and standard deviations was low for both the training and the test data.

The observations for Figure 10 were confirmed using analysis of variance, showing a significant difference in average training accuracy as the number of layers increased; that is, the network was trained more accurately as the number of hidden layers rose. Mean training accuracy increased as more nodes were added, but appears to reach an asymptote at approximately 30 to 40 nodes (Figure 10). The other statistically significant relationship involved the interaction between the number of layers and number of nodes. It appears that training accuracy improved as more layers were added to the network and as the number of nodes increased. However, training accuracy reached an asymptote, or even decreased, above approximately 30 to 40 nodes.

There was a statistically significant difference in the mean test accuracy for networks of one, two, or three layers, with the one-layer network producing lower accuracies when compared to two- and three-layer networks. Mean test accuracy also differed significantly as the number of nodes changed. The interaction effect between the number of layers and number of nodes per layer was also associated with a significant difference in mean test accuracy.

As more layers are added to the network, the more complex network allows the data to be modeled more accurately. Also, as more nodes are added, the training accuracy rises, while the test accuracy decreases. In order to understand this behavior, consider the situation with few hidden nodes. The connection weights of the hidden nodes vary with a large magnitude, as the error is propagated backwards by the neural-network. That is, a change in node weight tends to undo the previous change to the node. This causes the neural-network weights to oscillate wildly, reducing the possibility that a point of minimum error is reached. Because the neural network cannot reduce the remaining error, the average training accuracy remains low. As the number of nodes increases, the total error in the network falls, and the training accuracy increases.

The above results, obtained using analysis of variance, were confirmed with the nonparametric Kruskal-Wallis test. A significant difference in the median training and test accuracy was obtained for one-, two-, or three-layer networks. There is also a significant difference between the percentage of correct training and test cells for different number of nodes per hidden layers (for $p < 0.01$).

A related phenomena is that the two-layer neural network reaches a maximum training accuracy more rapidly than the one-layer network (Figure 11). In other words, the asymptote of the training accuracy curve for the two-layer network is achieved after about 2000 epochs, compared with approximately 5000 epochs for the one-layer network.

At a more practical level, a GIS analyst trying to decide on a suitable number of nodes per hidden layer may see that Figure 10 has the highest percentage of correctly classified training patterns at approximately 21 to 30 hidden nodes, but the test data accuracy is highest at approximately 11 to 20 nodes. The evidence here suggests the number of hidden nodes should be approximately 20 for one hidden layer, in order to maximize test accuracy while achieving a reasonable training accuracy. More hidden nodes are required for two- and three-layer networks; 20 to 30 nodes per layer maximizes test accuracy while achieving a high training accuracy. However, it should be emphasized that the results obtained suggest general trends, and the use of other data sets may cause the algorithm to perform differently.

Number of Epochs (Iterations)

As the number of iterations (or epochs) increases during neural-network training, the total system error becomes lower (Figure 12).

An experiment in which the number of epochs varied is shown in Figure 13, for the total system error (Figure 13a), training accuracy (Figure 13b), and test accuracy (Figure 13c). The total system error decreases as the number of iterations increases (Figure 13a), while the training accuracy improves as the number of processing cycles increases (Figure 13b). There is no obvious relationship between accuracy of

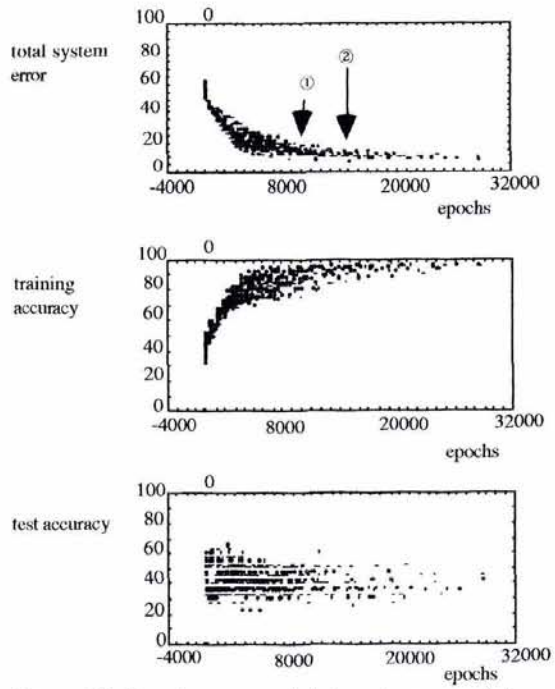


Figure 13. Epochs versus total system error (a), training accuracy (b), and test accuracy (c).

the test data and the number of epochs or total system error (Figure 13c).

The network appears to become "overtrained" as the number of epochs increases, a phenomenon typical of iterative optimization procedures (e.g., Goldberg, 1989). Overtraining occurs when training data, which are already well modeled by the algorithm (for example, point ①on Figure 13), continue to be iterated through the model; that is, the number of epochs continues to increase for the network. When unknown (test) data are presented to an overtrained network, the accuracy of predictions decreases (Figure 13c). Overtraining (or generalization) is caused by the network memorizing the input/output pairs, and becoming less able to generalize between similar input-output patterns (Haykin, 1994).

To formally test for overtraining, the data set was subdivided into epoch ranges of 10,000 to 14,000 epochs and above 14,000 epochs. Above approximately 10,000 epochs, total system error became asymptotic (Figure 13a), and, when the number of epochs increased above 14,000 (point ② on Figure 13), overtraining apparently occurred because training accuracy increased, while test accuracy decreased. Stated formally, the null hypothesis is Ho: $\eta_1 = \eta_2$ versus the alternate hypothesis Ha: $\eta_1 < \eta_2$, where η_1 is the median percentage of correct training cells for epochs in the range 10,000 through to 14,000, and η_2 is the median percentage of correct training cells for greater than 14,000 epochs. The Mann-Whitney U test rejected the null hypothesis at $p = 0.01$; thus, we conclude that training accuracy increases as the number of epochs increases. A similar null hypothesis was constructed for the test data. The null hypothesis was rejected at $p = 0.00001$; therefore, we conclude that the test accuracy becomes significantly lower as the number of epochs increases. Therefore, there is significant evidence that overtraining occurs; as the number of epochs increases, the accuracy of training continues to increase, but the test accuracy decreases.

It is important that the network is able to classify new patterns correctly with respect to the training patterns. A number of methods have been proposed to stop a neural network once it begins to overtrain (Haykin, 1994). For example, in the cross-validation procedure, the available data are divided into a training and a test data set. The training data are further split into a "training" set (to estimate the model) and a "validation" set (to evaluate the performance of the model). Overtraining shows up as reduced accuracy (performance) in the validation set. A method proposed by Haykin (1994) to detect overtraining involves monitoring the validation data set, and noting when the classification performance fails to improve by a user specified amount (e.g., 0.5 percent). At this point, the learning rate is reduced, and the neural network continues to iterate until the performance again fails to improve. After the size of the learning rate falls below a user specified threshold, network training is halted.

In the experiments reported here, the neural network executed until the total system error was reduced to a user specified level. This allowed the full behavior of the neural network with GIS and remotely sensed data to be examined. If the network is halted using stopping rules such as in the cross-validation method, other artifacts may be introduced into the performance of the network. For example, is the point of overtraining, as defined by the stopping algorithms, really a point of overtraining, or is it a local minimum on the error surface? In other words, use of stopping rules aimed at preventing overtraining, may cause other problems, such as not reaching the actual minimum on the error surface. However, other methods for stopping the network, such as using total system error as a criterion to "stop" the network, may also lead to sub-optimal classification performance through poor generalization. To date, no global method has been suggested for "stopping" the network in an optimal manner, such that the "best" balance between training accuracy, test accuracy, and system error is achieved.

Total System Error

The total system error is inversely correlated with the percentage of correct training data (correlation coefficient of -0.70 at $p < 0.01$; Figure 14), because the training data are iteratively used to reduce system error (see Equation 3). However, system error is not correlated with test accuracy. A GIS or image processing analyst should not use system error as the only criterion for determining the success of a classification, as it may lead to poor generalization.

Learning Rate and Momentum

Learning rate is analogous to the distance along the error surface traveled in a single epoch (Figure 15), so that the smaller the learning rate, the smaller the changes in the weights of the network at each epoch (Kosko, 1992; Haykin, 1994). If the learning rate is too large, the network may become unstable and oscillate across the error surface. Momentum is a term added to the learning rate to incorporate the previous changes in weight with the current direction of movement in the weight space (Rumelhart and McClelland, 1986; Kosko, 1992). In other words, inclusion of the momentum term avoids wild swings across the error surface, while allowing the system to learn faster.

Figures 16a and 16d show, respectively, the change in training and test data accuracy, as the learning rate increases from 0.1 to 0.9 and momentum is simultaneously varied from 0.1 to 0.9. In Figures 16b and 16e, data with a momentum of 0.1 are subset from Figures 16a and 16d. Similarly, Figures 16c and 16f are plots of data with momentum equal to 0.9. No obvious trends were apparent, except that a high (greater than 0.8) momentum coupled with a high learning rate (also greater than 0.8) appears to have lower training and test accuracies. This was tested formally by a null hypothesis, Ho:

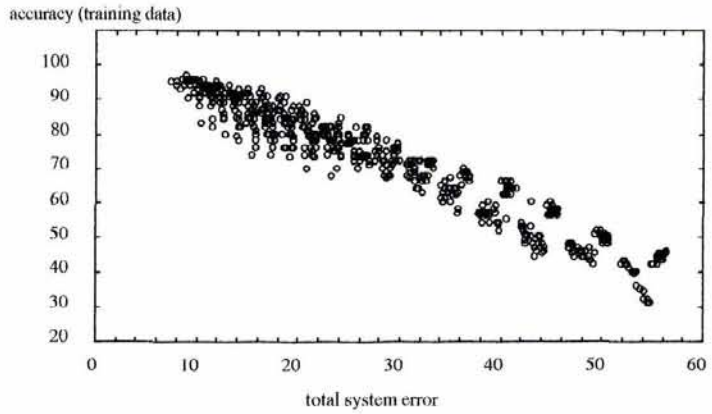


Figure 14. Total system error remaining after the neural network has executed against the percentage of correct training data.

$\eta_1 = \eta_2$, versus the alternate hypothesis Ha: $\eta_1 < \eta_2$, where η_1 is the median training accuracy for experiments with a learning rate greater than 0.8 and a momentum greater than 0.8, and η_2 is the median training accuracy for experiments with a learning rate less than 0.8 having an "exclusive or" relationship with momentum less than 0.8. The null hypothesis was rejected at $p < 0.00001$, so it is concluded that the combination of a high learning rate and momentum reduce the accuracy of training. A similar formal test constructed for the test data did not reject the null hypothesis.

Training accuracy (Figure 17a) and test accuracy (Figure 17b) appeared little changed as the momentum tended towards 1.0, and the learning rate was held constant at 0.2. A null hypothesis Ho: $\eta_1 = \eta_2$, versus the alternate hypothesis Ha: $\eta_1 \neq \eta_2$, where η_1 is the median training accuracy for a momentum greater than 0.8, and η_2 is the median training accuracy for a momentum less than 0.8 was tested using the Mann Whitney U test. The null hypothesis was not rejected (at $p = 0.05$) for training accuracy; a similar null hypothesis for the test data was also not rejected. Thus, training and test accuracy are unaffected by high momentum values. Thus, momentum allows faster learning (Rumelhart and McClelland, 1986), but does not increase test and training accuracy. An explanation is that the momentum constant (Equation 9) restricts oscillations in the network weights to the change in weight used in the previous epoch. Therefore, large changes in network weights are filtered out; the total system error decreases to the point it would have reached if the momentum term had not been used, while the training and test accuracy are not increased. Backpropagation methods based on the conjugate-gradient method appear to require fewer epochs than the standard backpropagation methods used here because the solution path across the error surface does not follow a zig-zag (Haykin, 1994). However, the conjugate-gradient method is computationally expensive to implement (Haykin, 1994).

A lower learning rate should require a greater number of epochs (to reach a minimum total system error), because the number of "steps" over the error surface will be larger (Figure 15). In Figure 18, the learning rate is plotted against epochs for momentum equal to 0.2, 0.5, and 0.8; points occurring within the lowest quartile of the system error are drawn on the left of Figure 18, and from the highest quartile on the right. More epochs are required at low learning rates; this was confirmed using the Mann Whitney U test, where a

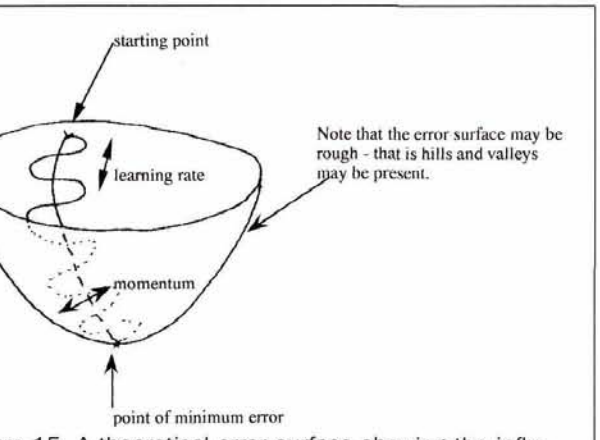


Figure 15. A theoretical error surface showing the influence of learning rate and momentum.

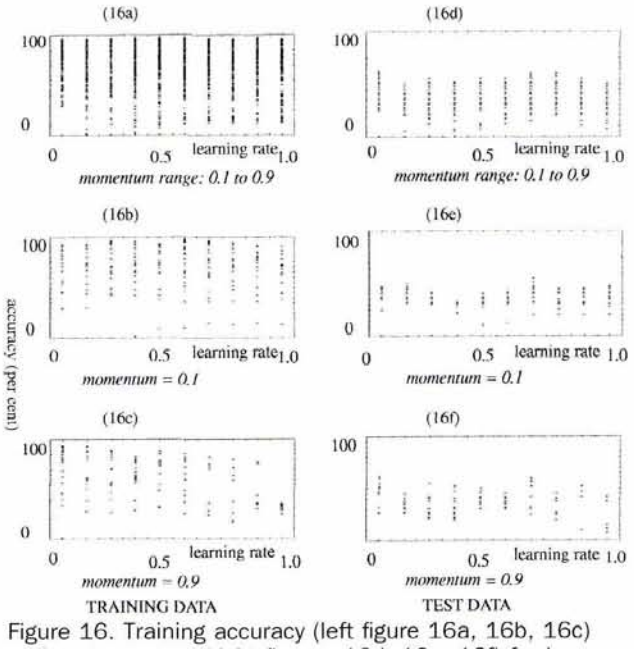
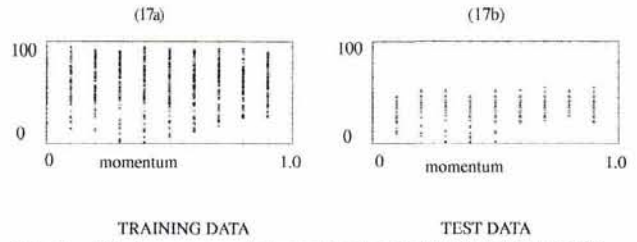


Figure 16. Training accuracy (left figure 16a, 16b, 16c) and test accuracy (right figures 16d, 16e, 16f) for learning rate 0 to 10.0 and different momenta.



17. Training accuracy (a) and test accuracy (b) for learning rate held constant at 0.2 and momentum varied from 0 to 1.0.

null hypothesis was formally tested, that is, Ho: $\eta_1 = \eta_2$ versus the alternate hypothesis Ha: $\eta_1 \neq \eta_2$, where η_1 is the median training accuracy for a learning rate greater than 0.5, and η_2 is the median training accuracy for a learning rate less than 0.5. The null hypothesis was not rejected for $p < 0.05$.

Target Values

Target values refer to the values the analyst assigns to each output node. The neural network adjusts the weights of connecting nodes by minimizing the system error; the aim is to calculate output nodes (o_{pk}) as close as possible to the target values (t_{pk}).

If the target values (t_{pk}) are set towards the high (or extreme) end of the range (i.e., 0.00 and 1.00, respectively), rather than the low (or negligible difference) end of the range (0.45 and 0.55), it is expected that the network will need to cycle through more epochs to minimize the system error (Figure 19). A null hypothesis Ho: $\eta_1 = \eta_2$ was tested using the Mann Whitney U test, versus an alternative hypothesis Ha: $\eta_1 > \eta_2$, where η_1 is the median number of epochs at target value pairs of [{0.01, 0.99} {0.05, 0.95} {0.10, 0.90} {0.15, 0.85} {0.20, 0.80}], and η_2 is the median number of epochs at

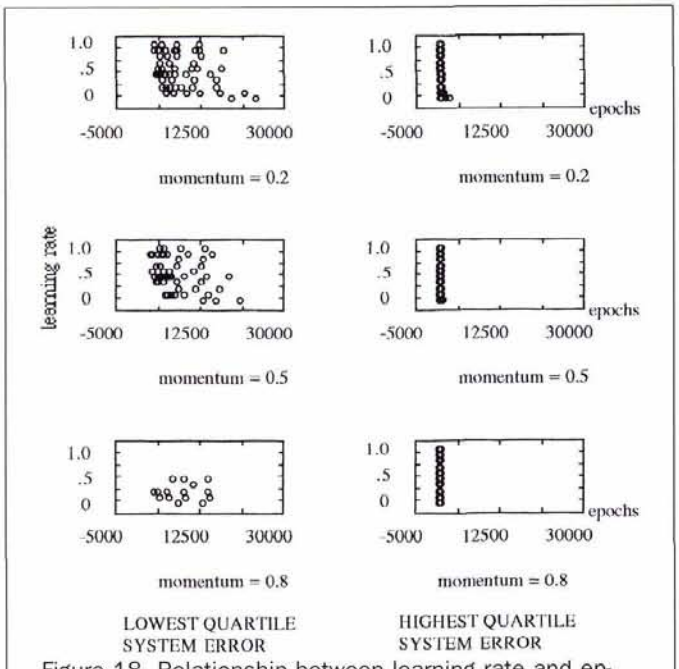


Figure 18. Relationship between learning rate and epochs in response to changing momentum, for data with a low and a high total system error.

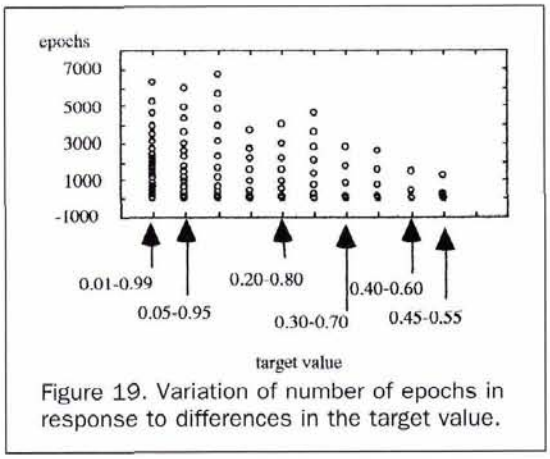


Figure 19. Variation of number of epochs in response to differences in the target value.

target value pairs of [{0.25, 0.75} {0.30, 0.70} {0.35, 0.65} {0.40, 0.60} {0.45, 0.55}]. The null hypothesis was rejected at $p < 0.002$, so concluding that more epochs are required for high (extreme) target values.

Training accuracy apparently decreased from target value pairs {0.1, 0.99} to {0.45, 0.55} (Figure 20). This observation was confirmed using the Mann Whitney U test hypothesis ($p < 0.00001$), so it is concluded that training accuracy is higher for extreme target values. A possible explanation is that, because fewer epochs are required with target values of {0.45, 0.55}, the network does not cycle through enough epochs to adequately train itself.

When test accuracy is plotted against target values, the relationship appears erratic (Figure 21). There were no statistically significant differences between the median test accuracy for different target values.

Training and Test Accuracy

Training accuracy increases in a generally linear relationship with test accuracy until approximately 30 percent, before reaching an asymptote (Figure 22). This may be an overtraining response by the network. As stated above, it is important that an analyst does not use training accuracy as the sole criterion to indicate success in modeling GIS and remotely sensed data with a neural network.

Manipulating the Starting Weights

How robust are neural networks to variation in starting node weights? A "successful" experiment was chosen, which had

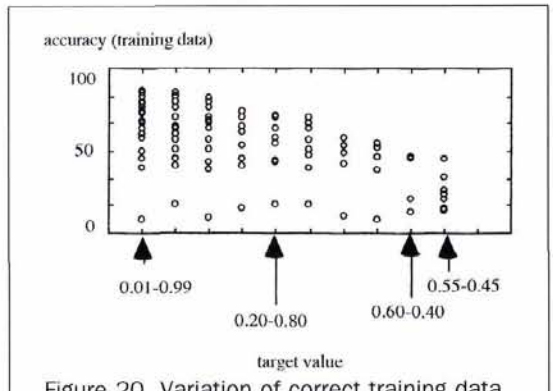


Figure 20. Variation of correct training data in response to differences in the target value.

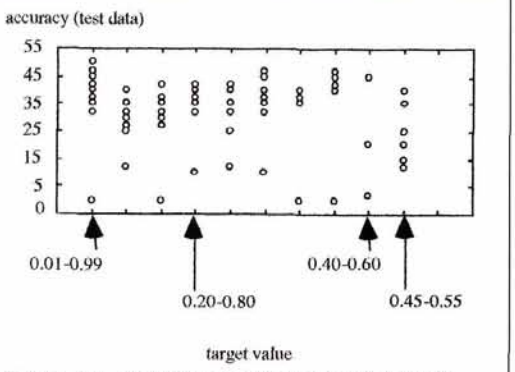


Figure 21. Variation of correct test data in response to differences in the target value.

high training and test accuracy, as well as a classification map which visually appeared to have the classes in the correct positions. To limit overtraining, a stopping rule was used, based on the point of decrease in the test data set. The network was trained on the full training set, and the generalization performance of the resulting network was measured on the test data set. The network parameters were noted (Table 2).

The network parameters (e.g., number of learning patterns, number of nodes, number of layers, learning rate, momentum, etc.) were held constant, except that the starting weights were randomly adjusted by ± 5 percent. The stopping rule was applied to the network. The five new maps were visually different (Plates 1b to 1f), and had a large variation in training and test accuracies (Table 3).

It is surprising that different results were produced when starting weights are randomized; the lack of robustness may worry an analyst. Another concern is the wide range of accuracies resulting from varying the network parameters (Figure 23). The choice of network parameters (e.g., number of hidden nodes, number of hidden layers, etc), as well as the initial weights of the nodes, is critical to the success of neural network applications.

Discussion

Equivocal is one word to sum up our experiences using the backpropagation algorithm for classifying eucalypt forest vegetation from GIS and remotely sensed data. The oft quoted advantages of neural networks, including "...the ability to take incomplete data and produce approximate results. Their parallelism, speed and trainability..." (Obermeier and Barron, 1989), were negated by the variable and unpredictable results generated. Undoubtedly, the neural network did work, and behaved in a manner which was understandable by an expert analyst, but the adjustments and fine tuning required of the input parameters would deter many users.

The results cited in this paper were generated by varying user defined parameters, including the type (and form) of data input to the network; the number of input, hidden, and output nodes; the (desired) total system error; the number of data patterns the neural network uses to learn with; the learning rate and momentum; and the node weights. The possible combinations of these parameters are large, and there is little information in the literature to guide an analyst about the optimal values at which to set these parameters with remotely sensed and GIS data. As shown in this paper, there are heuristics which may be derived, but these general rules will depend on the quantity, quality, and format of the data input to, and output from, the neural network.

Many results were surprising, for example, the total system error remaining high, even after many epochs. This may be due to a number of factors, but a likely candidate is the backpropagation algorithm being caught in "spurious local minima" on the system error surface (Aleksander and Morton, 1990).

It may be anticipated that stopping rules applied to prevent overtraining (Haykin, 1994) would result in a better balance between high training accuracy and good generalization (i.e., high test accuracy). However, such stopping rules will not solve the problem of "spurious local minima" on the system error surface.

TABLE 2. PARAMETERS FOR THE "SUCCESSFUL" EXPERIMENT

Parameter	Value
Number inputs	13
Number outputs	5
Number layers	4
Learning rate	0.20
Momentum	0.10
Number training patterns	150
Number of test patterns	40

TABLE 3. TRAINING AND TEST ACCURACIES FOR FIGURES 23a TO 23f

Figure Plate	Training data correct (%)	Test data correct (%)
23a 1a	93	42
23b 1b	93	47
23c 1c	97	45
23d 1d	90	50
23e 1e	96	50
23f 1f	92	55

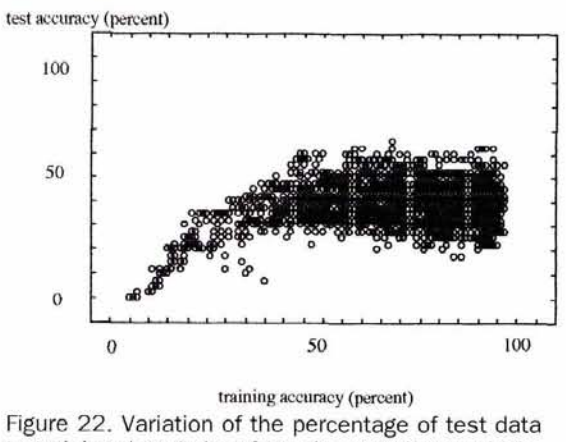


Figure 22. Variation of the percentage of test data to training data (taken from the experiment which held momentum constant).

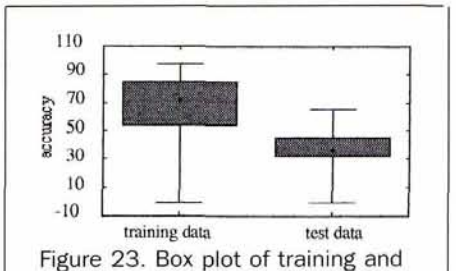


Figure 23. Box plot of training and test accuracies from all experiments.

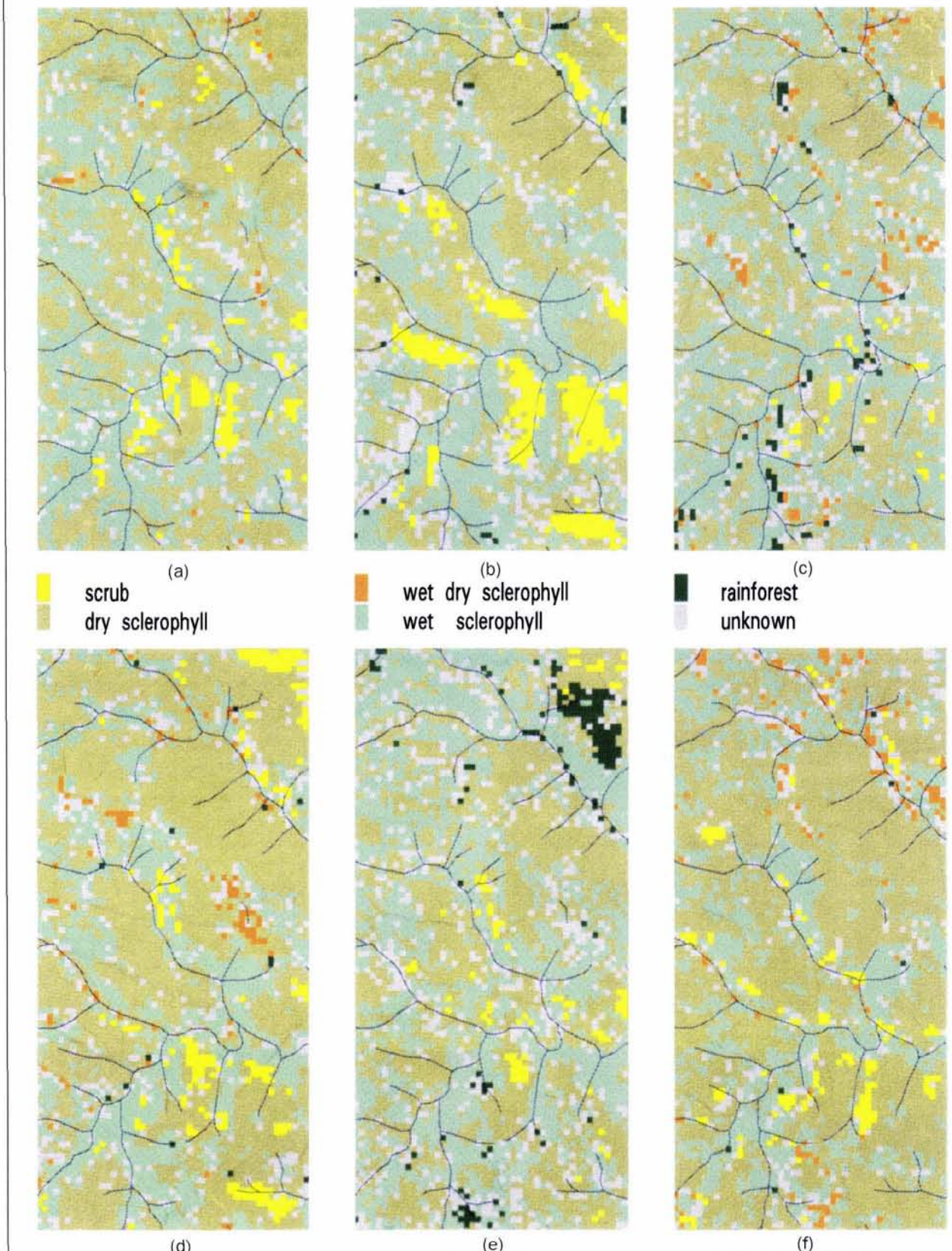


Plate 1. Effect of randomly varying the starting weights by ± 5 percent on the final classification.

Another surprising result is the very different visual appearance of the output classifications when the starting weights were randomly varied. Conventional classifiers, such as maximum likelihood, do not suffer from this limitation. At first glance, this seems a serious shortcoming of neural networks. In reality, the neural network may be reflecting the underlying uncertainty of the classification; the different appearance of the maps in Plate 1 indicates that mapping accuracy is low (also see Table 3). It is the complexity of setting up, and tuning of, the neural network "black box" that appears to limit its usefulness; put simply, using a maximum-likelihood classifier is much easier.

It took up to 48 hours to execute an experiment on a Sun Sparc 10 workstation; in contrast, experiments were completed in minutes on a Thinking Machines CM-5 computer with 32 processing nodes. Nonetheless, computational expense may be a problem with large (operational) data sets.

Some of the problems experienced in this study may be attributable to the data set being unsuited for a neural network (or even other classifiers such as maximum likelihood), due to its complexity and size. Most comparable experiments which have mapped forests note this difficulty. For example, Civco (1993) commented that "*...this initial neural network design is inadequate to achieve fine distinction...*" between coniferous trees, wetlands, and water. Civco (1993) adds that "*...this confusion between relatively low-reflecting coniferous trees and similar water and wetland features has been observed before in traditional maximum likelihood classifications.*" Obviously, the training examples input to the neural network must inherently contain the information to be modeled; otherwise, the relationships between the independent (input) and dependent (output) data will not be inferred.

The problems with neural networks are that they require good training data sets to yield a reliable result, and the large number of parameters make them difficult to use. Why, then, are they used? First, neural networks can identify subtle patterns in input training data, which may be missed by conventional statistical analyses. Second, neural networks are non-linear, and therefore may handle complex data patterns. Third, neural networks are able to take a specific set of input data and generalize a solution set, which will give the correct answer for unknown input patterns which are similar to, but not identical to, the input data. Finally, neural networks have great potential when used with field plot data, as the "information" content of the data may be "extracted" by the neural network automatically. This eliminates the need for specialists to analyze and model information of interest. In other words, the neural network may extract information from the data set that the specialist does not glean. Another advantage is that continuous, near-continuous (e.g., scanner rasterized data), and categorical data can be input without violating model assumptions.

In summary, we believe that the neural network backpropagation algorithm will probably not become a significant classification and analysis tool for GIS and remotely sensed data when implemented as a pure neural network. Where relationships are obvious in the data set, simpler algorithms such as maximum likelihood are probably more appropriate. However, neural networks may be a useful adjunct to other classification techniques, which utilize the advantages of neural networks, while minimizing their disadvantages. Of particular interest are techniques which combine expert systems and rule based methods (Skidmore, 1989) with neural networks.

Acknowledgments

The Australian Research Council and Genasys II Pty Ltd supported this research. Several staff members at State Forests of NSW facilitated the project: Mr. David Loane provided access to GIS data; Dr. Rod Kavanagh and Mr. Doug Binns provided plot data; field staff at Eden, particularly Mr. Bob Bridges, provided maps and local expertise; and Dr. Hans Drielsma encouraged collaboration. Mr. Philip Tickle and Dr. Roger Hnatiuk of the Australian National Forest Inventory assisted in identifying available data across the region. Field data were collected by Julie Delaney and Fiona Watford. Professor Peter Burrough of the University of Utrecht, The Netherlands, provided valuable comments on a draft of the paper. The work was partly undertaken during a study leave at the University of Utrecht, The Netherlands, and support from UNSW and the Organisatie voor Wetenschappelijk Onderzoek (NWO) (The Netherlands) is acknowledged. The thoughtful contribution of two anonymous referees is gratefully acknowledged; their corrections to the first draft, as well as general comments, led to the design of new experiments which added interesting information to the paper.

References

Aleksander, I., and H. Morton, 1990. *An Introduction to Neural Computing*, Chapman and Hall, London.

Anderson, J.R., E.E. Hardy, J.T. Roach, and R.E. Witmer, 1976. *A Land Use and Land Cover Classification System for Use with Remote Sensor Data*, U. S. Geological Service Professional Paper 964, U.S.G.S. Washington, D.C.

Baur, G.N., 1965. *Forest Types in New South Wales*, Forestry Commission of N.S.W., Sydney, Australia.

Bridges, R.G., 1983. *Integrated Logging and Regeneration in the Silvertop Ash-Stringybark Forests of the Eden Region*, Research Paper 2, Forestry Commission of New South Wales, Sydney, Australia.

Civco, D.L., 1993. Artificial neural networks for land-cover classification and mapping, *Int. J. Geographical Information Systems*, 7(2):173–186.

Fergusson, C.L., R.A.F. Cas, W.J. Collins, G.Y. Craig, K.A.W. Crook, C.McA. Powell, P.A. Scott, and G.C. Young, 1979. The Upper Devonian Boyd Volcanic Complex, Eden, New South Wales, *Journal of Geographical Society of Australia*, 26:87–105.

Fitzgerald, R.W., and B.G. Lees, 1992. The application of neural networks to the floristic classification of remote sensing and GIS data in complex terrain, *Proceedings of the ISPRS*, Washington, D.C.

Goldberg, D., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Massachusetts.

Haykin, S., 1994. *Neural Networks: A Comprehensive Foundation*, Macmillan, New York.

Hepner, G.F., T. Logan, N. Ritter, and N. Bryant, 1989. Artificial neural network classification using a minimal training set: Comparison to conventional supervised classification, *Photogrammetric Engineering & Remote Sensing*, 56(4):469–473.

Keith, D.A., and J.M. Sanders, 1990. Vegetation of the Eden Region, South East Australia: Species Diversity and Structure, *Journal of Vegetation Science*, 1:203–232.

Kosko, B., 1992. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, Englewood Cliffs, New Jersey.

Obermeier, K.K., and J.J. Barron, 1989. Time to get fired up, *Byte*, (August):217–224.

Omatu, S., and T. Yosida, 1991. Pattern classification for remote sensing using neural network, *International Joint Conference on Neural Networks in Singapore*, pp. 653–658.

Pao, Y.-H., 1989. *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, Massachusetts.

Parikh, J.A., J.S. DaPonte, M. Damodaran, A. Karageorgiou, and P. Podaras, 1991. Comparison of backpropagation neural networks and statistical techniques for analysis of geological features in Landsat imagery, *SPIE - Application of Artificial Neural Networks II*, 1469:526–538.

Richards, J.A., 1986. *Remote Sensing-Digital Analysis*, Springer-Verlag, Berlin.

Rumelhart, D.E., and J.L. McClelland, 1986. *Parallel Distributed Processing*, MIT Press, Cambridge, Massachucetts.

Skidmore, A.K., 1989. An expert system classifies eucalypt forest types using Landsat Thematic Mapper data and a digital terrain model, *Photogrammetric Engineering & Remote Sensing*, 55(10): 1449–1464.

Skidmore, A.K., and B.J. Turner, 1988. Forest mapping accuracies are improved using a supervised nonparametric classifier with SPOT data, *Photogrammetric Engineering & Remote Sensing*, 54(10):1415–1421.

Wilson, J.M., 1991. Back-propagation neural networks: A comparison of selected algorithms and methods of improving performance, *Proceedings of the 2nd Workshop on Neural Networks*, pp. 39–46.

(Received 11 January 1995; accepted 18 April 1996; revised 17 May 1996)