J Geograph Syst (1999) 1:3–22

Journal of Geographical Systems

Improving neural network performance on the classification of complex geographic datasets

Mark Gahegan, Gordon German, Geoff West

Department of Geographic Information Science, Curtin University of Technology,
P.O. Box U1987, Perth, Western Australia 6845, Australia
(e-mail: mark, gordon, geoff@cs.curtin.edu.au)

Abstract. Neural Networks are now established computational tools used for search minimisation and data classification. They offer some highly desirable features for landuse classification problems since they are able to take in a variety of data types, recorded on different statistical scales, and combine them. As such, neural networks should offer advantages of increased accuracy. However, a barrier to their general acceptance and use by all but 'experts' is the difficulty of configuring the network initially.

This paper describes the architectural problems of applying neural networks to landcover classification exercises in geography and details some of the latest developments from an ongoing research project aimed at overcoming these problems. A comprehensive strategy for the configuration of neural networks is presented, whereby the network is automatically constructed by a process involving initial analysis of the training data. By careful study of the functioning of each part of the network it is possible to select the architecture and initial weights on the node connections so the constructed network is 'right first time'. Further adaptations are described to control network behaviour, to optimise functioning from the perspective of landcover classification. The entire configuration process is encapsulated by a single application which may be treated by the user as a 'black box', allowing the network to the applied in much the same way as a maximum likelihood classifier, with no further effort being required of the user.

Key words: Classification, neural networks, G15

JEL classification: C88, C63, C45, C44

1. Introduction

Since the 1980's an enormous amount of literature has been generated on the topic of neural networks (e.g. Pao 1989; Freeman and Skapura 1991; Fischer 1994). They have been successfully applied to geographic and remotely-sensed data in a number of different studies (e.g. Bischof et al. 1992; Kamata and Kawaguchi 1993: Civco 1993). Despite this, their uptake as tools for geogra-

phy and remote sensing has been slow, with some researchers concluding that the problems encountered may outweigh the advantages (e.g. Skidmore 1995). However, others (e.g. Benediktsson et al. 1990) have argued that the distribution-free nature of neural networks allows them to exceed the accuracy of more conventional tools.

The work presented describes part of a research project to develop neural network tools that are easy to apply, but without sacrificing performance (in terms of accuracy and efficiency). The network package described tackles the classification problem; specifically, where a large and disparate input vector is being used to characterise (label) a dataset into a number of discrete, known classes. Such problems are common in the spatial sciences. In practice, the setup of a classifier 'net' typically involves a good deal of expertise and a large amount of experimentation. A study of the function of the various network components, as they relate to the classification problem, has brought to light various strategies for automating the configuration.

Some aspects of network configuration are presented. Results are given along the way to demonstrate the effects of the techniques described on a publicly available dataset. The entire classifier, including the dataset and relevant documentation, is available on the internet for those interested in trying it out or evaluating it against other classifiers (http:www.cs.curtin.edu.au/~gisweb/donnet/) The authors are keen to hear feedback from those who do so.

1.1 Classification and the dataset

The classification of areas of the Earth's surface in terms of land use, vegetation cover, soil type, etc. is often a necessary step prior to importing the data into a geographic information system. The final product is usually in the form of a chloropleth map, where each map element is assigned exactly one label. As such, the classification process is a discrete one; either an element belongs exclusively to a certain class or it doesn't. However, it is also possible to construct mappings where the ownership of a certain pixel is shared among several classes, giving rise to a probabilistic categorisation of the data (e.g. Foody 1996). This paper is concerned with the former approach, and neural network implementations for the latter will be discussed in a separate paper.

Discrete classification is the transformation of a set of *attributes* associated with a map element into a single class label. These attributes may represent many different themes, for example Landsat TM imagery, geological data, hydrological properties and so forth. Supervised classification is the process of classifying a dataset into a number of pre-defined discrete classes, based on some known data → class relationship (the ground truth). This is in contrast to unsupervised classification, where the classifier itself determines the output classes to be constructed. This research concentrates specifically on supervised classifiers.

Supervised classification procedures generally consists of the following three stages:

1. The *training phase*. In supervised learning, the classifier is trained to recognise certain combinations of attribute values as identifying a particular class, by providing a small number of sites within the area of interest for

which the class is known in advance. This *training set* should contain a representative sample of all the desired output classes.
2. The *verification phase*. Verification is the process of determining the success of the training phase, and hence the likely accuracy of the classifier when applied to unseen data. There are several methodologies employed for the verification of supervised classifiers. Quite often, the training set is simply represented to the trained classifier and the accuracy determined. This is, of course, not verification in the true sense, as all the samples in the set have already been seen by the classifier during the training phase. However, often there is just not enough samples available in the dataset, with associated ground truth, to facilitate the production of an additional sample set. Alternatively, a holdout regime might be employed, where one or more samples are held back during each training epoch and the entire set used for verification. However, the preferred method (and the one used in this research) is to produce a statistically independent set, the validation set, from the data (hence no member of the validation set is used in the training phase) and assess the accuracy of the classifier on this. This assessment is normally reported as the percentage of samples correctly classified. This can be calculated as simply the total number correctly classified, (Percentage Correctly Classified, or PCC), or as the mean of the percentage correctly classified per class (the Average Normalised Response, or ANR). The PCC figure is more often reported; however it does not take into account the real-world situation, where class sizes may vary enormously, so that the ANR should be the preferred figure. In this research, we present both figures, allowing direct comparison with other works which might quote a PCC figure. Validation results can also be presented as a confusion matrix (see Table 4 later) which shows the performance of the classifier across all output classes. Errors of commission, errors of omission, and Kappa statistics (Congalton 1991) can all be calculated straightforwardly from this matrix. To improve reporting accuracy, each experiment may be conducted a number of times on the same dataset, using a different division of training and validation data each time. The results presented here are an average obtained over a number of experiments[1].
3. The *classification phase*. The entire dataset is presented to the classifier, which then produces a classified map based on the relationships learnt during the training phase.

The data used in this paper is from the Kioloa dataset (Lees and Ritman 1991), the latest version of which will shortly be available as a NASA Pathfinder reference site. It describes a coastal region in New South Wales, Australia. The aim of the classification is to differentiate the various floristic species occurring in a highly variegated landscape. The region contains a small amount of rainforest, intermixed with eucalypt tree types, cleared land and coastline. As Fig. 1 shows, very little differentiation of forest is possible using Landsat TM data alone. Consequently, the dataset also contains geology, slope, aspect, flow accumulation and height coverages.

With such a large amount of ancillary information the exercise could be considered as species habitat classification as opposed to image classification.

[1] Care must be taken when comparing the results presented by different authors, as the methods of calculating success vary enormously.

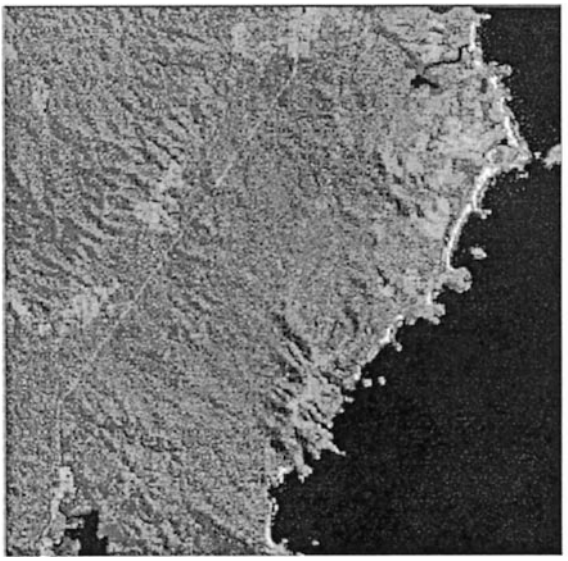

Fig. 1. Landsat TM scene fragment, using false colour (bands 2, 4 and 5).

In all, ten different input layers are available, some of which are ordinal (e.g. aspect, geology) whilst others are more quantitative (e.g. flow accumulation, Landsat TM). All data is in raster format on a common $30 * 30$ metre grid. Nine target classes have been identified and are characterised by a set of 1704 training points; in which individual pixels are labelled to represent the dominant vegetation cover, by field observation. Unlike agricultural landuse classification, the training sites do not represent contiguous regions in larger target objects, but are instead isolated and 'random' samples.

This dataset was selected because (i) it is publicly available, allowing others to compare results (Prechelt 1994) (ii) it presents a 'hard' classification problem (Lees 1994) and (iii) it contains highly diverse data that would be difficult to combine using a more conventional approach such as a maximum likelihood classifier (MLC) (e.g. Lillesand and Kiefer 1979).

1.2 Neural networks as supervised classifiers

Within the family of artificial neural network architectures, the MLP (or more formally the Multi-Layer Perceptron) can function as a highly parameterised, non-linear supervised classifier which "learns" by means of some form of cost minimisation, based on a given set of target values. Unlike many statistical classifiers such as the MLC, no prior assumption is made regarding the statistical distribution of the data; rather, the MLP learns a unique distribution for each dataset. Hence the often-used term *distribution free*.

In classification problems, the network attempts to produce the correct class labels for each of the input attribute vectors. Evidence (E) consists of a finite number (n) of observation vectors (X), each consisting of p pieces of evidence pertaining to a single location, $X = (x_1, x_2, \ldots, x_p)$. Given a set of q

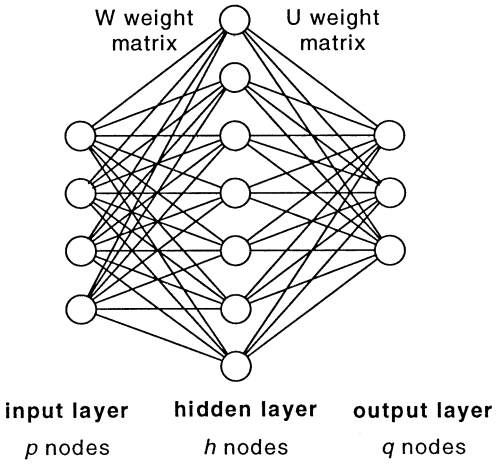

Fig. 2. A simplified network architecture.

mutually exclusive output classes $C = \{C_1, C_2, \ldots, C_q\}$, the network provides a mapping from each X to either a single class (the most dominant value) as: $\Gamma : E \rightarrow C$, or to the power set of classes as: $\Gamma : E \rightarrow 2^C$. The former 'winner takes all' approach is taken here; that is, the output is discrete. A diagram of a typical network is shown in Fig. 2.

Each neuron (or node) functions as an independent computational unit and behaves in the following manner: (i) all inputs to the node are summed, (ii) the summed value is passed through a non-linear activation function to give an output value and (iii) the output value is passed on, via a weighted connection, to the next layer of nodes. Typically, a non-linear activation function such as the sigmoid function is used, of the form

$$y_i^* = \frac{1}{1 + e^{(y_i+\theta)/\phi}}$$

where θ represents a threshold, or bias, and ϕ is the gain of the function, which in most implementations is held constant. The weighted connections, more commonly called weights, have their values constantly changed during the training phase, as the network attempts to map the input data to the desired output classes.

1.3 Overview of network structure

The configuration of a neural network remains a difficult problem to which a general solution and methodology has yet to be fully determined. Whilst there are any number of packages, architectures and searching strategies available (e.g. Cho and Kim 1993; Baum and Haussler 1989), the choice of network architecture and the mechanics of configuration and training are largely 'discovered' by experimentation. Such experimentation typically involves large amounts of time and requires an in depth understanding of network function.

By contrast, the commonly used maximum likelihood classifier (MLC) requires only brief user interaction to configure.

An earlier paper by the authors (German and Gahegan, 1996) describes a neural network 'black box' classifier (called DONNET–Discrete Output Neural Network), that can be used in much the same way as an MLC, in that it is robust and easily configured. It has the advantages that the input dataset may be of arbitrary complexity and that classification accuracy appears to remain stable (and above that obtained by an MLC). The correct architecture is determined from the data by the package, removing the normal "trial-and-error" approach required when using most other neural network packages. This ease-of-use also improves the repeatability of the classification process and is one of the main advantages of the DONNET package. Other variations from the standard MLP are presented in Section 3. A brief overview of the characteristics of the DONNET classifier are presented here, with some of the more pertinent points explained in detail in the next few sections:-

1. All nodes have one or more inputs and one output.
2. There is one input layer, one hidden layer and one output layer of nodes.
3. The hidden and output layer nodes map their inputs to their output via a non-linear activation function. The input layer simply passes its input to its output.
4. Weighted connections link each node in a layer to all others in the next layer (see Fig. 2).
5. The number of nodes in the input layer, p, is equivalent to the number of elements in the input data vector (?In the Kioloa dataset used here there are four bands of Landsat TM data, one geology layer, one hydrology layer and four surface morphology layers, so $p = 10$.).
6. The number of nodes in the output layer, q, is equivalent to the number of target classes to be recognised; that is, each output node represents a class label ($q = 9$ for this dataset).
7. The number of hidden layer nodes, h, is equivalent to the number of pairwise linear discriminant functions needed to separate the data into q classes. This is explained further in the next section.
8. The initial starting value for the weights are derived from the data, rather than starting from some random point in weight-space. This helps to assure convergence and speeds up training.
9. The node bias is modelled as an additional weight into each node for computational simplicity, but is mathematically equivalent to a variable threshold in the activation function.
10. A modified Scaled Conjugate Gradient algorithm is used for the minimisation of the cost, or objective function. All parameters are selected by the algorithm, removing the need for user involvement (Benediktsson et al. 1993; Moller 1993).
11. A winner-takes-all (WTA) strategy is employed in determining the class membership at the output stage i.e. the node with the highest output value labels that particular input pixel as belonging to its representative class.

2. Network learning

One choice that the user must make concerns the number of nodes required in the hidden layer. Too few nodes may not be able to separate out all desired

classes; too many may result in failure to converge in a reasonable time. The non-linearity of the input/output mapping can make it difficult to see the relationship between the network's nodes and weight connections and the separating out of the classes from the input data in the attribute space. This mapping of the activity in the network's w-dimensional weight space, where w can be many hundreds, to the p-dimensional attribute space ($p = 10$ for this data set) can be more easily grasped if one considers the classification problem from the point of view of a discriminant function classifier.

Rather than guess or experiment to find a suitable number of neurons for the hidden layer (h), an assumption is made regarding class separability. Specifically, it is assumed that each pair of output classes is separable by a single hyperplane, so h represents the number of pairwise discriminant functions needed to separate out the q classes. For the nine output classes in the Kioloa dataset the number of hidden nodes is therefore:

$$h = \binom{9}{2} = 8 + 7 + \cdots + 1 = 36.$$

Each hidden layer node is responsible for the positioning of a single hyperplane in *attribute* space. The dimensionality of the hyperplanes is determined by the number of weight connections at the input to the node, which is fixed by p. That is, each node h_i controls the position of a hyperplane separating a pair of classes in a p-dimensional *attribute* space. During the training phase, each hyperplane is positioned so as to reduce the classification error between two particular classes (see Fig. 5, shown later). Hence the attribute space is divided into "semi-classified" regions and it is the function of the output layer to combine these regions in such a way as to produce the final classification. Due to the non-linear nature of the node activation functions and the high level of connectivity between the hidden and output layers, this is far more than a simple summing or subtracting of regions in the classical set theory sense, with levels of *fading* and *mixing* being allowed amongst the semi-classified regions (Dunne and Campbell 1994).

In practice, results indicate that pairwise separation is tenable for many different types of geographic data, and in some cases gives an over-parameterisation of the problem which can be optimised later by net 'pruning'. There are, however, situations where this assumption breaks down where the classes are not easily separable given the training data, as is the case where the class boundary is convoluted. In this case, further improvement requires either a hyperplane of increased dimensionality (see later) or possibly the addition of further hyperplanes.

From the viewpoint of computational efficiency, it is desirable to initialise the net with weight values close to the final solution, so that training time might be reduced. In a three layer network, two sets of weights must be initially configured, termed the U and W matrices (see Fig. 2). A reasonable starting position is given by assuming that the classes are *linearly* separable and using the Fischer's group of linear pairwise discriminant functions (Dunteman 1984) as a basis for the starting network weights[2]. The procedure for configuring the W matrix from the training data is then:

[2] In fact, the entire network can be run as a linear discriminant classifier and gives a normalised classification accuracy (PCC) of around 48% on the Kioloa dataset.

For each hidden node $k(k = 1 \ldots h)$, calculate the discriminant function $LD_{i,j}$, separating *each pair* of classes i and j as:

$$LD_{i,j} = \frac{M_i - M_j}{\frac{1}{2}\Sigma_i + \frac{1}{2}\Sigma_j}$$

where M_i = mean vector for class i and Σ_i = covariance matrix for class i. Compute the grand mean (GM) of i and j, followed by θ, as the inner product of GM and $LD_{i,j}$. The elements of the vector $LD_{i,j}$ are the weights feeding into the k^{th} hidden layer node and θ is the bias of the k^{th} hidden layer node.

The W matrix is now known, so too are the input values, the required output values (the target) and the activation functions. The output from the hidden layer nodes and the required inputs to the output layer nodes can hence be determined, from which the correct U matrix can be found. The procedure is:

Fit a weights file to the net with the above calculated values for the W matrix and random values for the U matrix. Run the net for one iteration to produce the hidden layer outputs (a matrix X of size $p \times h$).

Determine the matrix B (of size $p \times q$) as the values expected at the *inputs* of the output layer nodes for the given targets (if the output activation function is sigmoid, an input value of -1.0 will give an output approaching 0.1; $+1.0$ will give an output approaching 0.9). Each row of B represents an expected vector of node inputs for a particular training vector and each column represents one of the output nodes. The U matrix is then calculated as the inner product of X^{-1} and B.

This approach to network architecture, configuration and training appears to be robust on the datasets tested so far, with classification accuracies of 70.79 PCC and 51.44 ANR obtained for the Kioloa dataset.

3. Further improvements to network architecture and training

Experiments have been conducted into various ways of refining the architecture and configuration to further improve on classification accuracy. These include (i) forced learning (ii) appropriate scaling of the data (iii) an alternative cost function and (iv) hyperplane distortion. Table 1 shows the incremental improvements obtained by applying these refinements to the basic network described in Section 1.3 (in terms of PCC and ANR over ten separate classifications). The following subsections detail these refinements and the results more fully. Notice that the best results from the training phase do not equate with the best results from validation, due to over-training of the network. Indeed, the results obtained by training can be improved still further to over 90% PCC, simply by allowing more iterations. However, the validation results will drop off accordingly. This problem of overfitting the data is one that the user needs to be aware of and has been discussed at some length in the literature (e.g. Sarle 1994; Skidmore 1995). However, it should by viewed in terms of misuse of a tool, rather than a failing of the methodology.

Table 1. Classification results for various enhancements

Net Type	Training		Validation		
	PCC	ANR	PCC	ANR	#Iterations
MLP (standard)	83.3	72.5	70.97	51.44	2000
MLP (scaled)	82.79	74.00	70.26	52.43	500
+ forced learning	84.07	74.75	66.12	54.82	500
+ randomised input	83.49	75.38	65.72	55.00	500
+ modified activation	87.52	74.51	68.37	57.20	120
+ new cost function	83.13	73.38	72.61	60.37	120

3.1 Forced learning

By including a suitable enhancement vector, the error during the training phase for vectors of a particular class can be given a higher precedence than other classes. These values are then used to scale the cost function and the derivatives depending on which class the particular input vector is supposed to be assigned to. The effect is to concentrate the learning (movement of the separating hyperplanes in attribute space) on certain class pairs.

The methodology for calculating these values is as follows:

Fit the net with the weights as derived from the linear discriminant functions (see above).

Run the dataset once through the net to generate an output classification and construct a confusion matrix based on this output.

From this confusion matrix, calculate normalised class success; i.e. the number of correctly classified samples (the values on the main diagonal of the confusion matrix) divided by the respective class size.

We intuitively require an inverse relationship between the normalised class successes and any required scaling of the cost function. The data is therefore fitted to a smooth negative exponential function with an offset to allow for both inhibitory and excitatory biases. So, the normalised class success vector is transformed via a function:

$$enhvalue_i = ae^{-bx_i} + c$$

where x_i is the normalised class success for class i.

Values for the coefficients a, b and c have been found empirically. The results from testing on several datasets have suggested the best generalised performance is obtained when

$$enhvalue_i = 3e^{-5x_i} + 0.35.$$

3.2 Data preparation

The input data itself can be scaled to improve network training time. Scaling ensures that all input nodes respond over a similar dynamic range. This ef-

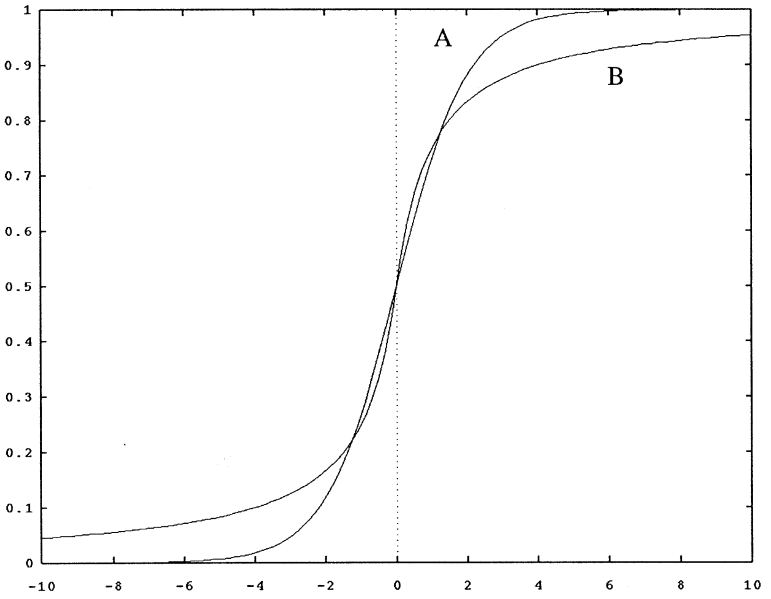

Fig. 3. Two possible sigmoidal activation functions. A shows the more traditional form, B has extended *non-linear* regions.

fectively reduce the area of weight-space to be searched, since ordinarily the network has to account for the differences in signal magnitude that occur between the various input layers by 'learning' their relationship one to another. Scaling also removes any initial bias towards a particular layer based arbitrarily on he relative magnitude of recorded data values. With a reduction in the *weight* space to be searched, the number of training epochs required for net convergence is decreased. A reasonable solution is attained in around 120 iterations on this dataset, with further training giving only marginal improvements of under two percent. Without scaling, convergence takes around 1,000 iterations. In real terms, this represents a time reduction from about sixty minutes to around twelve minutes for the training task (on a 150 MHz Silicon Graphics Indy).

A further advantage is that there is a reduced chance of saturation of the hidden layer. The non-linear sigmoid activation function quickly saturates to a value of 1.0 or 0.0 as the node input approaches 1.0 or -1.0 (see Fig. 3). This prevents any further differentiation of the input value. With the high fan-in rate of the hidden layer nodes that comes with the use of large dimensional input vectors, saturation can readily occur, preventing convergence of the network. Scaling the inputs helps to ensure that the occasional high-valued weight vector (selected during the search of the weight space) will not saturate the nodes.

Presenting the training set in the same order at each iteration can lead to reinforcement of learning errors and overtraining. Consequently, the order of the input vectors is randomised after each presentation epoch, to alleviate this problem.

3.3 Activation function

The non-linearity of the activation function associated with a given hidden node acts on the hyperplane produced by that particular node's input weight connections. This can be considered as a more indeterminate positioning of the hyperplane (as opposed to the discrete positioning that would result from a simple step function), allowing an input vector lying close to the hyperplane to 'exist' in more than one class. The gain and bias of the sigmoidal activation function gives the network some control over indeterminacy (currently, in the DONNET classifier, like most neural net classifiers, only the bias is variable).

The sigmoid functions commonly used for nodal activation consist of an approximately linear region, bounded on either side with a non-linear region and finally a saturated region, where the output value remains constant for a change in input (see Fig. 3).

The standard sigmoid function defined earlier goes into saturation for any input values x, $x < 5$ or $x > 5$. By extending the non-linear regions (effectively allowing a greater swing in the input before saturation occurs), we allow a greater 'smearing' of the hyperplane position, giving the network a better chance of separating out input vectors that overlap in the attribute space. Consequently, DONNET allows the use of both the standard sigmoid function described earlier, or an approximate sigmoid given by

$$y^* = 0.5 + \frac{x}{(1 + \text{abs}(x)))}$$

and shown in Fig. 3 where A represents the standard sigmoid and B the modified version. This function increases the active region of the input before saturation. classification results, when compared to the standard sigmoid function, are presented in Table 2.

3.4 Cost function

The majority of neural network implementations use a least squares formulation for the cost, or objective function:

$$E = \sum_n \sum_k \frac{1}{2}(t_k - z_k)^2$$

where t_k is the target value expected at an output node k, z_k is the actual output of that node and n is the number of input vectors in the training set.

Table 2. Comparison of activation functions

Hidden layer sigmoid	Output layer sigmoid	ANR
normal	normal	56.37
increased gain	normal	58.16
normal	increased gain	59.50
increased gain	increased gain	60.37

This function not only incorporates the error at the output node of interest (z_i, where i is the class to which the input vector belongs) but also all the residual errors associated with the other output nodes. In effect, the separating hyperplanes are readjusted needlessly whilst the minimisation routine attempts to reduce these residual values as well, with no beneficial increase in the classification accuracy. Hence, for a winner-takes-all strategy, minimising the above function is not necessarily the most efficient way of selecting the correct weights and an alternative approach to constructing a cost function can be postulated.

One possibility is to reduce the penalty associated with the losing nodes and their associated errors. The idea here is that, so long as the correct node has the highest value (hence the input vector is assigned to the correct class), the network should not be penalised for small 'errors' in the output of losing nodes, *provided* they are below some error threshold (κ). The following algorithm implements this idea:

Calculate output node errors as per least squares formula (above).
If [winning node = correct class] AND [Σ(losing node error) $< \kappa \times$ number of classes] then
  reset the error of all losing nodes to zero.
If (winning node error $< \kappa$) then
  reset the error of the winning node to zero

This also requires a slight change to the calculation of the partial derivatives of δE, which are needed by the minimisation routine. This algorithm has so far produced moderate increases in the generality of the network, in as much as the classification accuracy remains about the same for the training set, but improves slightly for the validation set (see Table 1). Obviously, a value must be chosen for κ, within the range 0.0 to 0.2 (greater than this will allow overlap between the winning and loosing values). Figure 4 shows a plot of κ vs classification accuracy.

The experimental evidence suggests that a value of 0.015 provides a large

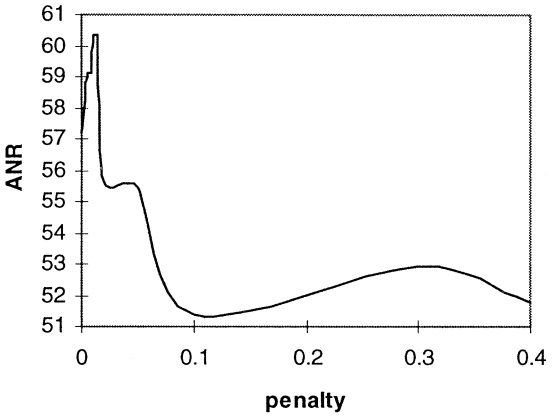

Fig. 4. Graph of cost function penalty versus Average Normalised Response (ANR).

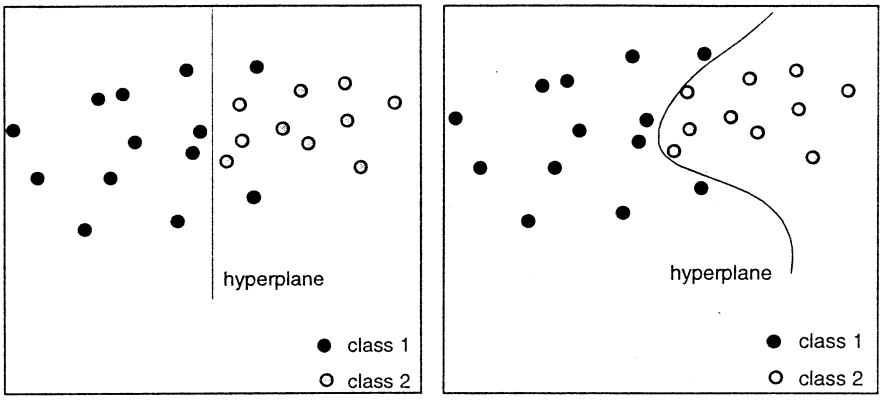

Fig. 5. Class separation using hyperplanes, contrasting the fitting abilities of standard and distortable hyperplanes.

enough buffer to stop the hyperplanes from being moved unnecessarily, whilst still giving enough of an inhibitory effect on the residuals.

3.5 Hyperplane distortion

If the data values in attribute space are reasonably linearly separable (between class pairs) then the classification accuracy should be high. In most of the experiments conducted so far this assumption has proved reasonable. Where the complexity of the class boundary is such that a single hyperplane cannot produce a satisfactory fit then some output classes may remain difficult to separate out. Ideally, careful selection of input datasets might avoid this problem altogether, but it is rarely the case that we can collect the ideal dataset that we require.

Where class overlap problems persist, and in the absence of additional data layers, the attribute space may be enhanced by borrowing a technique from the Functional Link Networks (FLN) proposed by Pao (1989). Each element of the input vector is additionally represented by one or more 'enhancement' nodes, whose value is a function of that element. In the case of where more than one node is added, the functions chosen should be ortho-normal to each other. The effect is to increase the dimensionality of the attribute space by the number of added nodes, but without actually adding in more data. This (hopefully) allows a better separation of output classes since the hyperplanes now have additional degrees of freedom, effectively allowing some distortion, as shown in Fig. 5.

In the current version of DONNET, enhancement is triggered when the decrease in classifier error per epoch is less than some user-defined value (typically 0.001). Training then continues on the whole enhanced network, but at a slower pace due the increase in attribute dimensionality. Two enhancement nodes e_1 and e_2 are added for each input node p_i. Given the value of p_i as x_i, e_1 and e_2 are initialised as $e_1 = \sin(x_i)$ and $e_2 = \cos(x_i)$. The weighted connections for these nodes are derived from the original weight, via the ortho-normal function value, scaled to the magnitude of the original weight.

Table 3. Comparison of various classifiers on the Kioloa dataset

Net Type	Training		Validation	
	PCC	ANR	PCC	ANR
MLP (DONNET)	81.13%	73.38%	72.61%	60.37%
MLP (standard)	45.7%	–	–	–
Maximum Likelihood	50.50%	–	–	–
Decision Tree	50.90%	50.57%	–	–

Table 4. Training class confusion matrix for enhanced network

	dry sclero-phyl	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**172**	5	6	5	14	0	2	0	0
E. botryoides	9	**24**	1	6	1	0	1	2	0
lower slope	7	2	**14**	4	4	0	3	1	0
wet E. maculata	27	0	3	**125**	8	3	2	0	0
dry E. maculata	12	2	1	22	**83**	1	0	0	0
rainforest ecotone	6	1	1	9	5	**43**	0	0	0
rainforest	6	3	0	4	1	2	**42**	0	0
cleared land	0	0	0	0	0	0	0	**111**	0
water	0	0	0	0	0	0	0	0	**332**
class total	204	44	35	168	121	65	58	111	332

Although the results (in Table 1) show some improvement, this is not an ideal method, since the adaptation currently affects the whole attribute space rather than just the poorly separable regions. Experimentation with a more focussed approach is under way at present. In this alternative strategy, the enhancement nodes are spawned only when the classification rate between two classes fails to pass a particular threshold, accompanied by little change in error per epoch. Similarly, the addition of two enhancement nodes is arbitrary (although based on observed improvements). Ideally, the number of enhancement nodes should in some way reflect the complexity of the class pair boundary.

4. Results

The fully enhanced network (summarised by the last row in Table 1) produces the best results in terms of the validation accuracy. Optimising performance based on the training set alone does not appear to produce classifiers that are generalisable. For comparison, Table 3 shows the results of a decision tree a standare MLP (Fitzgerald and Lees 1994) and an MLC (Fitzgerald 1995) on the same dataset. Table 4 shows the training set confusion matrix for the fully enhanced network and Table 5 the validation set confusion matrix. Bold figures represent the correct assignments. The resulting classified overlay is shown in Fig. 6. Full confusion matrices for some of the other modified networks of Table 1 are presented in the appendix.

Table 5. Validation class confusion matrix for enhanced network

	dry sclero-phy1	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**72**	3	5	7	15	0	0	0	0
E. botryoides	8	**9**	1	1	2	0	0	1	0
lower slope	5	4	**3**	1	1	1	1	1	0
wet E. maculata	16	1	0	**40**	17	7	2	0	0
dry E. maculata	8	0	0	14	**36**	2	0	0	0
rainforest ecotone	4	1	0	8	1	**16**	2	0	0
rainforest	2	0	0	4	3	2	**18**	0	0
cleared land	2	0	0	1	0	0	0	**52**	0
water	0	0	0	0	0	0	0	1	**165**
class total	102	22	17	83	60	32	29	55	166

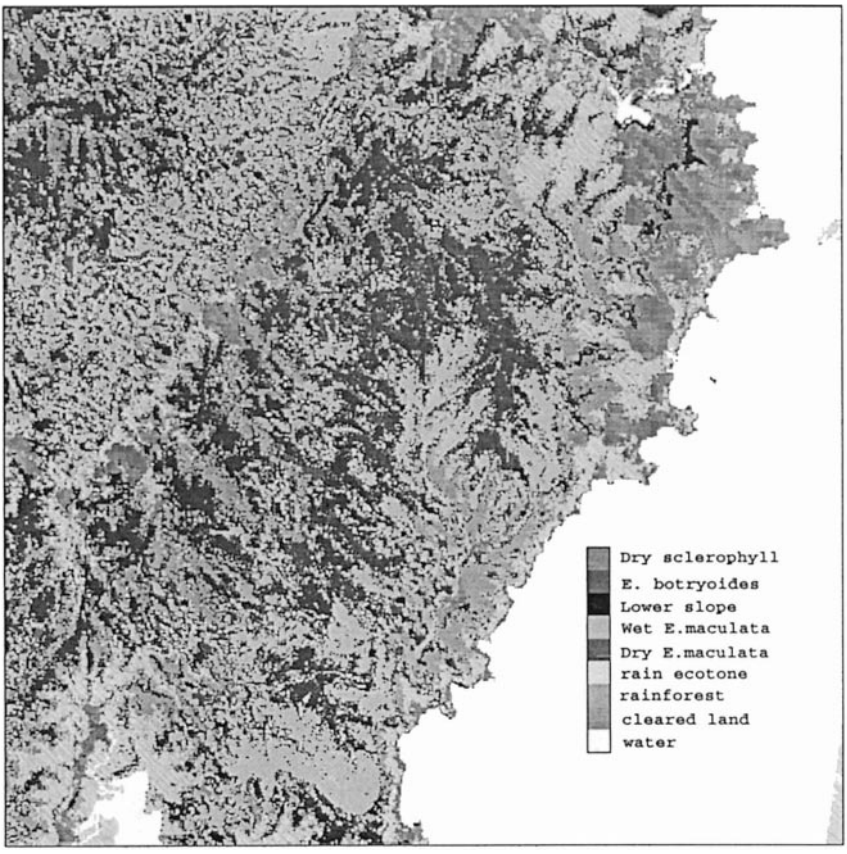

Fig. 6. Classified landcover theme produced by the fully enhanced network described.

5. Conclusions

The DONNET classifier has been developed as a stand-alone application, with a simple user interface. It automates many of the tasks and decisions that must ordinarily be tackled before a neural network may be used effectively and we hope will provide the functionality of neural networks to the non-expert user. As shown in the previous section, it will out-perform many other classifiers on complex GIS data.

DONNET has been exhaustively tested on various other geographic data-sets including agricultural scenes (German and Gahegan, 1996). Results are encouraging, in that the network has so far always converged on a good solution. No problems with saturation have been encountered.

There are still a number of improvements to be made. Specifically, the method of positioning hyperplanes could be improved by taking into account measures of within-class variance (rather like the MLC). This should give better results in the validation stage. Generally speaking, we intend to move to a more model based paradigm to facilitate training on more complex patterns and relationships.

Acknowledgement. Our thanks go to Dr. Brian Lees, Department of Geography, Australian National University, Canberra, for his provision of the data used in this paper.

Appendix

Following are some of the confusion matrices relating to the improvements of Section 3. For a particular improvement, both the training set and validation set matrices are shown.

Table A1. MLP with SCG (Section 2)

TRAINING SET	dry sclero-phy1	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**169**	2	2	14	11	1	4	1	0
E. botryoides	7	**28**	0	2	3	1	0	3	0
lower slope	12	1	**16**	0	2	0	3	1	0
wet E. maculata	21	2	1	**122**	12	6	4	0	0
dry E. maculata	14	3	0	15	**85**	3	1	0	0
rainforest ecotone	5	2	0	15	2	**37**	4	0	0
rainforest	6	0	0	6	1	1	**44**	0	0
cleared land	2	0	0	0	0	0	0	**109**	0
water	0	0	0	0	0	0	0	0	**332**
class total	204	44	35	168	121	65	58	111	332

VALIDATION SET	dry sclero-phy1	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**79**	0	0	8	9	2	2	2	0
E. botryoides	9	**0**	0	2	2	2	4	3	0
lower slope	13	0	**0**	0	0	1	1	2	0
wet E. maculata	14	0	0	**49**	11	4	5	0	0
dry E. maculata	14	0	0	10	**34**	1	1	0	0
rainforest ecotone	2	0	0	9	5	**14**	2	0	0
rainforest	5	0	0	5	2	1	**16**	0	0
cleared land	4	0	0	0	1	0	1	**49**	0
water	0	0	0	0	0	0	0	0	**166**
class total	102	22	17	83	60	32	29	55	166

Table A2. MLP with SCG, forced learning and randomised inputs (Section 3.2)

TRAINING SET	dry sclero-phy1	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**181**	1	1	9	7	2	2	1	0
E. botryoides	9	**28**	0	4	1	0	1	1	0
lower slope	10	2	**18**	1	2	0	1	1	0
wet E. maculata	25	2	1	**113**	17	7	3	0	0
dry E. maculata	15	2	0	17	**82**	4	1	0	0
rainforest ecotone	9	0	1	10	2	**42**	1	0	0
rainforest	7	1	0	6	1	1	**44**	0	0
cleared land	1	0	0	0	0	0	0	**110**	0
water	0	0	0	0	0	0	0	0	**332**
class total	204	44	35	168	121	65	58	111	332

VALIDATION SET	dry sclero-phy1	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**54**	12	12	6	10	2	6	0	0
E. botryoides	4	**7**	2	0	2	5	0	2	0
lower slope	1	4	**5**	2	0	2	1	2	0
wet E. maculata	12	2	4	**28**	18	18	1	0	0
dry E. maculata	7	3	1	7	**37**	4	1	0	0
rainforest ecotone	2	4	2	10	2	**12**	0	0	0
reinforest	1	3	1	1	2	5	**16**	0	0
cleared land	0	0	0	0	0	0	0	**53**	0
water	2	0	0	0	0	0	1	5	**160**
class total	102	22	17	83	60	32	29	55	166

Table A3. MLP with SCG, forced learning, randomised input and modified activation function (Section 3.3)

TRAINING SET	dry sclero-phy1	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**165**	6	3	14	9	1	5	1	0
E. botryoides	4	**34**	0	3	0	0	3	0	0
lower slope	2	0	**31**	1	1	0	0	0	0
wet E. maculata	22	0	3	**136**	4	2	1	0	0
dry E. maculata	14	0	0	3	**90**	3	1	0	0
rainforest ecotone	8	0	0	3	7	**47**	0	0	0
rainforest	4	2	0	2	0	0	**50**	0	0
cleared land	0	0	0	0	0	0	0	**111**	0
water	0	0	0	0	0	0	0	0	**332**
class total	204	44	35	168	121	65	58	111	332

VALIDATION SET	dry sclero-phy1	E. botry-oides	lower slope	wet E. mac-ulata	dry E. mac-ulata	rain-forest ecotone	rain-forest	cleared land	water
dry sclerophy1	**61**	4	7	13	15	0	1	1	0
E. botryoides	8	**8**	1	3	1	0	0	1	0
lower slope	6	3	**4**	1	0	1	1	1	0
wet E. maculata	14	1	1	**35**	19	11	2	0	0
dry E. maculata	9	0	0	14	**33**	4	0	0	0
rainforest ecotone	3	0	1	7	3	**14**	2	2	0
rainforest	3	0	1	3	3	1	**18**	0	0
cleared land	0	1	1	1	0	0	0	**52**	0
water	0	0	3	1	0	0	0	0	**162**
class total	102	22	17	83	60	32	29	55	166

References

Benediktsson JA, Swain PH, Ersoy OK (1990) Neural network approaches versus statistical methods in classification of multisource remote sensing data. IEEE transactions on Geoscience and Remote Sensing 28(4):540–551

Benediktsson JA, Swain PH, Ersoy OK (1993) Conjugate gradient neural networks in classification of multisource and very high dimensional remote sensing data. International Journal of Remote Sensing 14(15):2883–2903

Bischof H, Schneider W, Pinz AJ (1992) Multispectral classification of Landsat images using neural networks. IEEE Transactions on Geoscience and Remote Sensing 30(3):482–490

Cho S, Kim JH (1993) Feedforward Neural Network Architectures For Complex Classification Problems. Proceedings 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka92

Civco DL (1993) Artificial neural networks for landcover classification and mapping. International Journal for Geographical Information Systems 7(2):173–186

Congalton RG (1991) A review of assessing the accuracy of classifications of remotely sensed data. Remote Sensing of Environment 37:35–46

Dunteman GH (1984) Introduction to multivariate analysis. Sage Publications, New York, USA

Dunne R, Campbell N (1994) Some practical aspects of pruning multi-layer perceptron models applied to remotely sensed data. Research Report #94/06, Murdoch University, Western Australia

Fischer MM (1994) Expert Systems And Artificial Neural Networks For Spatial Analysis and Modelling. Geographical Systems 1:221–235

Fitzgerald RW (1995) Neural Networks: Successful Classification. GIS User, No. 13, pp 60–61
Fitzgerald RW, Lees BG (1994) Assessing The Classification Accuracy of Multisource Remote Sensing Data. Journal Remote Sensing Environment 47:362–368
Freeman JA, Skapura DM (1991) Neural networks: algorithms, applications and programming techniques. Addison-Wesley, New York, USA
Foody GM, McCulloch MB, Yates WB (1995) Classification of remotely sensed data by an artificial neural network: issues relating to training data characteristics. Photogrammetric Engineering and Remote Sensing 61(4):391–401
German G, Gahegan M (1996) Neural network architectures for the classification of temporal image sequences. Computers and Geosciences 9:969–979
Lees BG, Ritman K (1991) Decision tree and rule induction approach to intergration of remotely sensed and GIS data in mapping vegetation in disturbed or hilly environments. Environmental Management 15:823–831
Lees BG (1994) Decision Trees, Artificial Neural Networks and Genetic Algorithms for Classification of Remotely-Sensed and Ancillary Data. Proceedings, 7th Australasian Remote Sensing Conference, Vol. 1, Remote Sensing and Photogrammetry Association Australia, Floreat, Western Australia, pp 51–60
Lillesand TM, Kiefer RW (1979) Remote Sensing and Image Interpretation. Wiley New York
Kamata S, Kawaguchi E (1993) A neural network classifier for multi-timporal Landsat images using spatial and spectral information. Proceedings IEEE 1993 International Joint Conference on Neural Networks, Vol. 3, pp 2199–2202
Moller MF (1993) A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6:525–533
Pao YH (1989) Adaptive pattern recognition and neural networks. Addison-Wesley, Reading, MA, USA
Prechelt L (1994) A Study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice. Technical Report, Faculty of Informatics, University of Karlsruhe, Germany
Sarle W (1994) Neural Networks and Statistical Models. Proceedings 19th Annual SAS Users Group, No. 320, pp 1538–1550
Skidmore A (1995) Neural networks and GIS. GIS User, May–July 1995, pp 53–55