IMPROVING THE LEARNING ABILITIES OF A NEURAL NETWORK-BASED GEOCOMPUTATIONAL CLASSIFIER

Gordon German, Mark Gahegan and Geoff West

Geographic Information Science
Curtin University of Technology
Bentley, Western Australia 6102
ph +618 9266 3145
fax +618 9266 2819
email: *gordon@cs.curtin.edu.au*

KEY WORDS: Neural networks, classification

ABSTRACT

The classification of complex geographic datasets remains a long-standing problem. From a geocomputational perspective, the many new techniques provided by computer science can offer some significant performance enhancements, but also give rise to a new set of problems. The use of neural networks for classification is one such example, where separating boundaries in attribute space are constructed from hyperplanes produced by the hidden-layer nodes. In an earlier work, the authors have shown how the Boundary Misclassification Rate (BMR) matrix can be used to analyse the functioning of the neural network classifier during the learning phase and help predict the classification ability for a given task.

We now extend this to show how the learning ability of the network can be optimised to the task at hand, by using the BMR matrix to spawn additional hidden-layer nodes as required, during or prior to, the learning process. Importantly, this process can be both automated or user-controlled, so that the accuracy of the classifier can either be maximised across all classes, or specific class separations can be targeted for attention, as meets the user's requirements.

Results on a real world GIS dataset are presented and compared to results obtained previously without this optimisation; these show encouraging improvements in performance.

1. INTRODUCTION

Modern technology has vastly expanded the range of measurable attributes that can be gathered and analysed from a particular geographic space, especially with the improvements in remote sensing technology and the increase in available platforms (e.g. Wilkinson *et al.*, 1995). In order to make sense of these increasingly numerous (and varied) types of measurements, some form of data reduction and/or grouping is necessary that maintains the focus of inquiry set by the researcher. As such, classification is one of the more common transformations that data undergo. Simply put, the task is to produce a mapping,

$$\Re^p \xrightarrow{\Gamma(n)} \Pi^q$$

where p is the number of input variables, or geodata type attributes and q is the number of output variables, or classes. Unlike reduction techniques such as principle components analysis or canonical variate analysis, the goal of classification is to select output classes from a (often) smaller and different phenomenological domain (Π, the classification *scheme*) to that of the input attributes ($\Re$). These transformation models can be catagorised as unsupervised or supervised classifiers. In unsupervised classification, a particular Π, or scheme, is chosen by the classifier, normally based on some form of cluster analysis applied in $\Re$. In the case of supervised classification, of which our classifier is a representative, the scheme is chosen by the user and the classifier *learns* an approximation $\Gamma'(p)$ to the required transfer function $\Gamma(p)$. Hence the (common) scheme of ground cover type is often derived from an attribute domain that may comprise several bands of LANDSAT data, as well as ancillary data such as digital elevation models, rainfall, etc..

Supervised classification schema are popularly used by the GIS professional as an aid to the decision-making processes in disciplines such as land management, mineral exploration, environmental science and habitat modeling. The dominant forms of these classifiers are based on well-established statistical models, themselves derived from the theories of Baysian estimation (Mardia *et al.*, 1979). The maximum likelihood classifier (MLC) is an example and usually considered the better of these classifiers, at the expense of computational overhead (Rao, 1973). These classifiers, on the whole, try to estimate the transfer function $\Gamma(p)$ by modeling it as a known statistical distribution $\Phi[T:p]$ (T being the defining parameter set of Φ), which assumes a stochastic sampling process in selecting a learning, or *training set* in $\Re$, as well as the assumptions of the associated per-class probability density functions $\phi_i(x)$. The most common distribution used is the Gaussian, as in the MLC. Because of the requirement that a given $\phi(x)$ approximates the actual attribute population distribution for that class, these classifiers have problems with nominal or ordinal attribute data, as attribute values are assumed to be ranked when constructing $\phi(x)$. These classifiers also require a minimum number of samples in each representative class within the training set to allow a meaningful population distribution to be derived for the modeling of $\phi(x)$. The exact number is dependent on the number of degrees of freedom for that particular Φ and the dimensionality of the attribute space, p (Dunteman, 1976).

An artificial intelligence (AI) approach to classification has lead to the development of classifiers whose basis is computational rather than statistical in nature. Decision trees, genetic algorithms and artificial neural networks are all examples of this alternative approach (Lees, 1994). Although these classifiers can often themselves be modeled statistically (Sarle, 1994), it is important to note one fundamental difference; no statistical assumptions are made in the implementation of these models, other than the fundamental one that Euclidian distance has some meaning within the attribute space. As such, on the one hand, they do not

suffer from the limitations noted above, but on the other, they suffer from their own unique set of problems, as well as from the limitations of the classification philosophy itself. The authors, in previous papers (German *et al.*, 1997; Gahegan & German, 1996) have focused on artificial neural networks, specifically a variant of the multi-layered perceptron (MLP) coined DONNET (Discrete Output Neural NETwork) and addressed some of these problems, as well as issues of performance, as compared to the statistically derived classifiers. In this paper, we will address one further problem, that of building complex decision boundaries, which leads to either overall improved performance as measured on a given set of training or validation data, or to class-specific improvements, depending on the information required by the user.

2. Model Complexity - Hyperplane Analysis

2.1 Overview of DONNET - Previous Work

DONNET is a software simulation of an MLP, written to test the research propositions of an ongoing project at the Dept. of Geographic Information Science, Curtin University. It is available via the World Wide Web to other interested researchers (see address at the end of this article). Essentially, DONNET is an extension of the philosophical stance taken for the task-based MLP (Brieman *et al.*, 1984; Dunne *et al.*, 1992). Within this conceptual model, each node of the (single) hidden layer is responsible for the generation of a separating hyperplane within the p dimensional attribute space, whose *task* is to separate out one particular class (c_i) from one other (c_j). The output layer nodes and associated connections are then responsible for the amalgamation of one or more of these hyperplanes into *q-1* class boundary decision surfaces. In earlier work (Dunne *et al.*, 1993; German & Gahagan, 1996) it was shown that, assuming complete linear separability of the classes, $q \times (q\text{-}1) / 2$ hidden layer nodes are required to guarantee convergence on a solution for $\Gamma'(p)$. It was also noted that such a formulation works well, even when several of the classes are not linearly separable. This was shown (German *et al.*, 1997) to be due to the fact that often, one hyperplane could separate out more than one pair of classes (eg c_1:c_2 and also c_1:c_4), making other hyperplanes "redundant". These redundant hyperplanes were then used by the network to construct more complex piecewise linear decision surfaces that could be used elsewhere for non-linear cases. In the majority of applications, this provides excellent results and a comparison of DONNET's performance in relation to other AI and statistical techniques can be found in Gahegan & German (1996). However, in certain cases, where a majority of classes are *not* linearly separable from any other, performance can be poor. It is this situation we wish to address.

2.2 Using the BMR and Task Matrices to Determine Redundancy

The BMR (Boundary Misclassification Rate) matrix is described in German *et al.* (1997). It is derived from the class confusion matrix (e.g. Dunteman, 1976) and details the effectiveness of all pairwise class separations in terms of errors of omission and errors of commission (single-error boundaries), or both (dual-error boundaries). A standard task matrix is derived from the ouptuts of the hidden layer and shows the effective pairwise tasks being done by each hyperplane (or hidden-layered node). It is often used as a basis for pruning such networks (Brieman *et al.*, 1984; Dunne *et al.*, 1992). From this, a zero-confusion task matrix can be produced, as defined below. A worked example will show how redundancy can be inferred from these matrices.

We will use a dataset from the Kioloa area of New South Wales, Australia, which has been made available as a NASA pathfinder data-set through the Australian National University in Canberra (Lees and Ritman, 1991). There are 9 floristic-level classes to be delineated from 11 attribute layers (four of these represent nominal or ordinal data, four are Landsat TM bands) i.e. $q = 9$ and $p = 11$. The classifier is therefore initially set up with 11 input nodes, 36 hidden nodes (calculated from the formula given above) and 9 output nodes. This dataset is considered a "hard" classification problem, with a large overlap of class signatures in the data. Prior to training, the DONNET classifier is fitted with weights derived from Fisher's linear discriminant functions (Mardia *et al.*, 1979; German, 1995), rather than just randomly selecting weights. Table 1 shows the initial class confusion matrix of the classifier on the training set. Table 2 shows the associated BMR matrix, where positive figures represent errors of omission, negative figures are due to errors of commission and bold typed figures are dual-error figures. The dual-error figures are indicative of complex decision boundaries for which a single hyperplane may be inadequate for class separation (see Figure 1). Using the following notation:

- $\{T\}h_i$ is the set of all tasks which node h_i performs,
- $\{T_{min}\}h_i$ is the set of all tasks performed by h_i as well as, or better than, any other node,
- T_{hi} is the original, or primary task of h_i, as performed by that node,
- $\{T_j\}h_i$ is that task T_j performed by node h_i, that is also the primary task of node h_j,
- SCORE[T_{hi}] is the confusion with which h_i performs its primary task,

we can define the following levels of redundancy:
A node h_i is considered to make node h_j *partially* redundant if :

$$SCORE\left[T_{hj}\right] \geq SCORE\left[\left\{T_j\right\}h_i\right].$$

A node h_i is considered to make node h_j *completely* redundant if :

$$SCORE\left[T_{hj}\right] \geq SCORE\left[\left\{T_j\right\}h_i\right] = 0.$$

Nodes h_i and h_j are considered *compatibly* redundant if :

$$SCORE\left[\left\{T_{\min}\right\}h_i\right] = SCORE\left[\left\{T_{\min}\right\}h_j\right] = 0,$$

and both h_i and h_j are completely redundant. In other words, h_i and h_j are compatibly redundant if their primary tasks are performed with zero confusion by other nodes *and* they both perform the same set of zero-confusion tasks. These definitions will be used in conjunction with the zero-confusion task matrix. The importance of compatibly redundant nodes is this: given a set of compatibly redundant nodes, it is only necessary to maintain one in its original configuration, the others may be removed or moved elsewhere in attribute space without significant degradation of the performance of the original set of tasks.

Table 3 gives the hidden-layer zero-confusion task matrix. This matrix lists all nodes that perform tasks with zero confusion, i.e. perfect separation. The node number h_i is listed at the start of each row, followed by those tasks that it performs with zero confusion. There are several points to note from these tables:

1. As training has not yet been commenced, compatibly redundant nodes, as identified from the zero-confusion task matrix, are not to be pruned - they will be used by the

network elsewhere in $\Re^p$ for construction of non-linear class decision boundaries (more precisely, *piecewise linear* class decision boundaries).
2. It follows that training time could be significantly reduced by moving these redundant hyperplanes into the *areas of concern* prior to commencement of training. The areas of concern are those class pairs shown in bold in the BMR matrix.
3. Additional hyperplanes can be constructed and placed in the appropriate areas if there are not enough redundant nodes available, or more are required during training.

3. MANIPULATING HYPERPLANES

3.1 Positioning hyperplanes prior to training

The implementation of the above modifications requires some caution in the initial setup of the additional hidden-layer nodes. If analysis of the BMR and task matrices reveals the need for additional hyperplanes, or movement of redundant ones, they must be placed in position in attribute space so as not to perturb the current state of the network (in terms of the error, or cost function) too greatly. Further, new nodes and weights must be added so as to avoid having parallel hyperplanes within the attribute space, as the minimisation routines used by the network have problems distinguishing between these when calculating the partial derivatives necessary for weight updates. Let us return to the example analysis of the network with the Kioloa dataset.

Note that within the body of this text, quoted classification accuracy (%ANR) is a percentage based on the *validation* set, not the training set and is calculated as the *average* of the classification performance of each class, rather than a total. This avoids any misrepresentation due to large differences in class sizes, a common problem with real-world data. Performance on the training set, as well as overall percentages (%PCC), are quoted in the performance table (Table 7) at the end of this article, to allow comparisons with other classifiers.

First of all, it is necessary to identify areas that may require additional complexity (in terms of the class decision boundary). Looking at the BMR matrix of Table 2, we note that the decision boundaries at c_1:c_2 (10.4%), c_1:c_4 (12.4%), c_1:c_5 (10.7%), c_4:c_5 (17.8%) and c_4:c_7 (9.8%) are the major contributors to the total dual-error figure. They have resulted from a complex separating surface, an overlap of class data, or a combination of both, the implication being that these errors will not be significantly reduced by simple movement of the (single) separating hyperplane.

We can now use the zero-confusion task matrix of Table 3 to identify compatably redundant hyperplanes. Here we note that nodes 7, 14, 20, 25, 29, 32 and 34 are compatably redundant (node 21 is not, as its primary task, T_{21}, is not performed by the other nodes). This results in 6 redundant hyperplanes that can now be repositioned in proximity to the decision boundaries identified above from the BMR matrix. As an example, let us concentrate on the c_4:c_5 boundary (node h_{22} is primarily responsible for this task - T_{22}). The compatibly redundant hyperplane primarily associated with task T_7 (separating class 1 and 8) can be repositioned as follows:

1. Construct a line between the group centroids for class 4 and class 5.
2. Calculate the intersection point P between this line and the hyperplane associated with h_{22}.
3. Construct a new hyperplane H_A passing through P with some small offset angle α_1 (typically $\alpha_1 < 10^o$) from the primary hyperplane H_{22}.
4. Fit the coefficients of the polynomial describing H_A as input weights to the redundant node h_7.
5. If further hyperplanes H_B, H_C, ... are to be associated with this boundary, fit as above, but with new angles α_2, α_3, ... such that $\alpha_1 \neq \alpha_2 \neq \alpha_3$ etc.

(In actual fact, any reasonable method of calculating the hyperplane H_A could be used here, providing it positions the hyperplane within the error envelope associated with the decision boundary and ensures H_A is not parallel with H_{22}).

This process is repeated for each redundant node that is to be refitted. In this example, there are a total of 6 compatibly redundant nodes identified from the task matrix and 11 dual-error boundaries identified from the BMR. It is important to note that at this stage (prior to training), the size of the dual-error figures in the BMR matrix is *not* indicative of the complexity of the boundary. The BMR matrix only allows us to discriminate between dual-error and single-error boundaries. With this in mind, we will arbitrarily assign one redundant hyperplane (node) to each of the first 5 dual-errors. The U weights are recalculated as per German & Gahegan (1996) and a new initial classification performance figure of 48.03% is calculated for comparison with the original network (51.9%). This initial figure is now slightly lower, but we would expect this as we have "manually" placed the redundant hyperplanes, reducing slightly the sub-optimal fit that was given from the calculation via the linear discriminants. What we will expect to see is a convergence on a final error figure within fewer training epochs than that achieved with the original network.

3.2 Initial training

Both the original and the repositioned networks are now trained on the same dataset. The total network error per epoch is plotted for each in Figure 2. As expected, the repositioned network reaches a stable minimum in a significantly smaller number of epochs (approximately 60 as opposed to 180) and goes on to a slight improvement in classification of 61.2 % over the original's 60.8%. The class confusion matrix is shown in Table 4. There is no significant improvement if the network is allowed to train further (performance on the validation set peaks at 430 epochs at 62% - see Figure 3) other than overtraining. For any significant increase, we must look at increasing the dimensionality of the input data, or adding hidden layer nodes.

3.3 Adding hyperplanes

If the network is examined after 60 epochs, the BMR of Table 5 will result. As demonstrated above, a simple continuance of training will not lead to a significant improvement in classification, so we shall add further hidden-layer nodes before additional training. Note that the majority of error (in Table 5) is now due to dual-error boundaries, as the network has repositioned the available hyperplanes to the best of its ability, eliminating most single-error boundaries. From this BMR matrix, we can now select the c_1:c_3, c_1:c_4 and c_4:c_5 task separations as contributing to the majority of the dual-errors (tasks T_2, T_3 and T_{22}). We require additional hyperplanes for these decision boundaries, implying further nodes must be added to the hidden layer. Using the technique presented in Section 3.1, we now add

four more hyperplanes (nodes) to these areas of attribute space and retrain the repositioned network. In practice, the network is run for a few short iterations (we use 10 per hyperplane) between the addition of each node to optimise the output layer connections, thereby reducing the risk of moving too far away from the previous local minimum[1]. The resulting classification figure after training for a further 150 epochs (over the original 60) is now 67.3%, with the class confusion matrix for the training set shown in Table 6. (c.f. the standard network left to train for 200 + 150 epochs still only has a best classification rate of 61.2%).

4. CONCLUSIONS AND FURTHER WORK

The above techniques can be used to reduce training time, produce a better overall classification, as we have done here, or target specific classes for greater accuracy. For instance, in the Kioloa dataset, class 7 (rainforest) may be required to be delineated with greater accuracy than that of the other classes for the purposes of ecological study. In this case, the additional hyperplanes can be placed along the error envelope of the appropriate boundaries (eg. c_4:c_7, c_5:c_7 and c_6:c_7). The method could be extended to "freezing" hyperplanes that are giving adequate performance and then constraining the remaining hyperplanes to the area of interest in attribute space. One question that remains unanswered is how many additional nodes are feasible, or how far one persists with adding nodes during training. Gains in classifier accuracy decrease with additional complexity, to the point where further additions provide no further gain when tested on the validation set (for this dataset, an additional 2 nodes gives the peak figure of 68.6%). Further work could consider some metric to describe the class overlap complexity to be used by the network to determine the probable number of additional nodes required for a given level of accuracy.

With all these methods, it is important to measure the classification performance on a validation set, as we have done here, to avoid the trap of overfitting the network to the training data and losing generalisability. For a comparison, Table 7 shows the performance of DONNET, both modified and unmodified, with an MLC and a decision tree (C4.5) on the same dataset. Note the greater generalisation ability of the neural networks over the other classifiers (compare %ANR validation scores).

With prudent use, these classifiers can produce classification schemes with greater accuracy and without the limiting assumptions of the traditional statistical methodologies. Their further advantages lie in the ease with which disparate types of data (e.g. nominal and ordinal ancillary data, remote-sensed data etc.) can be combined, as well as the ability to model a distribution with relatively few examples.

---

[1] The network "learns" by searching for a minimum in some defined multi-dimensional error space (E^w, where $w >> p$), analogous to the weight space. So at any particular point of the training phase, the network has calculated and stored the last minima found and the directions required to get there. We do not want to invalidate this information by moving too far from this point, when we add an extra node and its weight connections.

FIGURE 1 : Reducing error at complex non-linear class boundaries. (a) Original single hyperplane used to model the decision surface. (b) Using two hyperplanes to model the same decision surface.

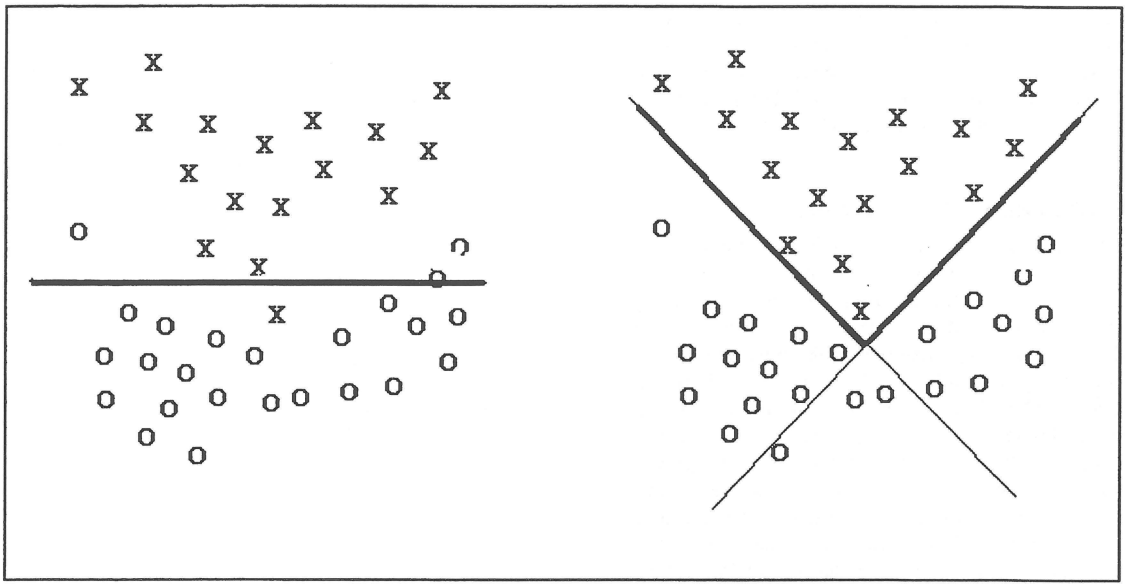

FIGURE 2 : Comparison of error for standard (f1_5) network and repositioned (f1) network

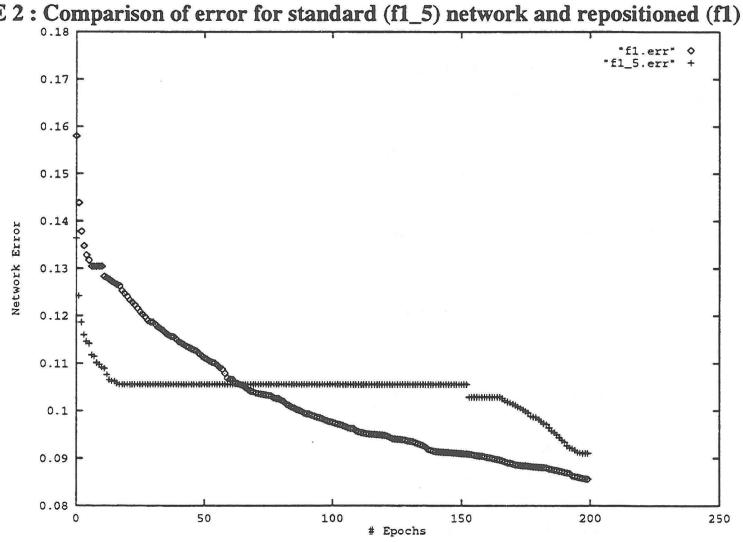

FIGURE 3 : Error for repositioned network, 0 ~ 1000 epochs

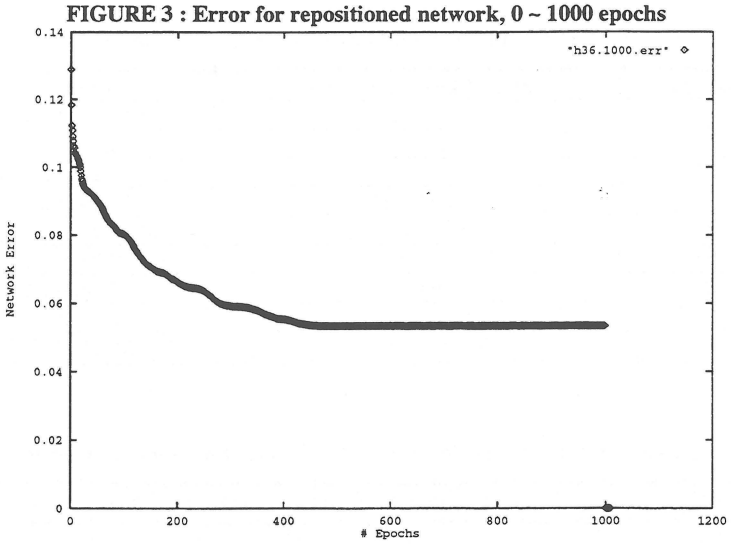

Table 1 : 9 Class Confusion Matrix (0 iterations)

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Totals
True 1	**162**	1	0	17	15	0	4	5	0	204
True 2	24	**0**	0	5	3	3	5	4	0	44
True 3	25	0	**1**	1	3	0	4	1	0	35
True 4	42	0	0	**97**	21	2	6	0	0	168
True 5	22	0	0	22	**73**	1	3	0	0	121
True 6	10	0	0	31	6	**17**	1	0	0	65
True 7	14	3	0	9	7	0	**25**	0	0	58
True 8	0	2	0	0	0	0	0	**109**	0	111
True 9	0	0	0	0	0	0	0	0	**333**	333

Table 2 : 9 Class BMR Matrix (0 iterations)

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Avg
Class1	-	**10.4%**	-9.7%	**12.4%**	**10.7%**	**7.3%**	+5.7%	**2.2%**	0%	**7.3%**
Class2	-	-	+2.2%	**2.5%**	-1.1%	-1.6%	**4.3%**	-1.1%	0%	**2.9%**
Class3	-	-	-	0%	-2.3%	-0.9%	-2.9%	-1.2%	0%	**2.4%**
Class4	-	-	-	-	**17.8%**	-11.8%	**9.8%**	0%	0%	**6.8%**
Class5	-	-	-	-	-	+1.0%	**3.0%**	0%	0%	**4.5%**
Class6	-	-	-	-	-	-	**4.7%**	0%	0%	**3.4%**
Class7	-	-	-	-	-	-	-	0%	0%	**3.8%**
Class8	-	-	-	-	-	-	-	-	0%	**0.6%**
Class9	-	-	-	-	-	-	-	-	-	**0.0%**

Table 3 : Zero Confusion Task Matrix for 9 Class Example (0 iterations)

Hyperplane	Primary Class Separation	Zero Confusion Tasks
6	Task 6 (c_1:c_7)	6,13,19,24,28,31,33,35
7	Task 7 (c_1:c_8)	7,14,20,25,29,32,34,35
14	Task 14 (c_2:c_8)	7,14,20,25,29,32,34,35
20	Task 20 (c_3:c_8)	7,14,20,25,29,32,34,35
21	Task 21 (c_3:c_9)	7,14,20,25,29,32,34,35
25	Task 25 (c_4:c_8)	7,14,20,25,29,32,34,35
29	Task 29 (c_5:c_8)	7,14,20,25,29,32,34,35
32	Task 32 (c_6:c_8)	7,14,20,25,29,32,34,35
34	Task 34 (c_7:c_8)	7,14,20,25,29,32,34,35

Table 4 : 9 Class Confusion Matrix for standard network (200 iterations)

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Totals
True 1	**183**	2	1	11	2	3	1	1	0	204
True 2	10	**24**	3	3	2	0	0	2	0	44
True 3	12	1	**15**	2	1	0	3	1	0	35
True 4	20	1	0	**125**	11	4	7	0	0	168
True 5	15	1	0	20	**83**	1	1	0	0	121
True 6	7	2	0	18	4	**33**	1	0	0	65
True 7	5	1	0	10	1	1	**40**	0	0	58
True 8	1	1	0	0	0	0	0	**109**	0	111
True 9	0	0	0	0	0	0	0	0	**333**	333

Table 5 : 9 Class BMR Matrix (200 iterations)

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Avg
Class1	-	**5.0%**	**7.4%**	**11.2%**	**9.0%**	**3.3%**	**2.7%**	**0.6%**	0.0%	**4.9%**
Class2	-	-	+5.6%	**1.7%**	**2.7%**	**2.4%**	**5.2%**	-2.3%	0.0%	**3.1%**
Class3	-	-	-	0.0%	-2.3%	+0.9%	-2.9%	-1.2%	0.0%	**2.5%**
Class4	-	-	-	-	**12.9%**	**10.7%**	**7.5%**	0.0%	0.0%	**5.5%**
Class5	-	-	-	-	-	**1.9%**	**3.0%**	0.0%	0.0%	**4.0%**
Class6	-	-	-	-	-	-	**3.6%**	0.0%	0.0%	**2.8%**
Class7	-	-	-	-	-	-	-	0.0%	0.0%	**3.1%**
Class8	-	-	-	-	-	-	-	-	0.0%	**0.5%**
Class9	-	-	-	-	-	-	-	-	-	**0.0%**

Table 6 : 9 Class Confusion Matrix for repositioned network + 4 extra hidden nodes (60 + 150 iterations)

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Totals
True 1	**191**	3	0	5	4	1	0	0	0	204
True 2	6	**31**	0	2	3	0	1	1	0	44
True 3	5	1	**25**	0	1	0	2	1	0	35
True 4	14	1	0	**146**	4	1	2	0	0	168
True 5	11	1	1	12	**95**	1	0	0	0	121
True 6	4	4	0	10	6	**41**	0	0	0	65
True 7	3	2	0	5	1	3	**44**	0	0	58
True 8	1	0	0	0	0	0	0	**110**	0	111
True 9	0	0	0	0	0	0	0	0	**333**	333

Table 7 : Performance of Classifiers on the Same Dataset. The MLC (Maximum Likelihood Classifier) figures are from Fitzgerald & Lees, 1993. The Decision Tree figures were produced with the package C4.5. DONNET A is the standard neural network. DONNET B is the modified network with 4 additional nodes in the hidden layer.

CLASSIFIER	*TRAINING SET*		*VALIDATION SET*	
	%PCC	*%ANR*	*%PCC*	*%ANR*
MLC	50.50			
Decision Tree	78.31	65.27	66.96	52.36
DONNET A	79.46	65.25	73.50	61.75
DONNET B	89.20	82.11	77.74	67.33

5. REFERENCES

Brieman, L., Friedman, J., Olshen, R. and Stone, C., 1984, *Classification and Regression Trees*, Wadsworth International.

Dunteman, G. H., 1984, *Introduction to multivariate analysis.* Sage Publications, New York, USA.

Dunne, R., Campbell, N. A. and Kiiveri, H. T., 1992, Task Based Pruning, *Proceedings of the 3rd Australian Conference on Neural Networks,* ACNN'92, Melb., pp. 166-169.

Dunne, R., Campbell, N. A. and Kiiveri, H. T., 1993, Classifying high dimensional spectral data by neural networks, *Proceedings of the 4th Australian Conference on Neural Networks,* ACNN'93, Melb.

Gahegan, M., German, G. and West, G., 1996, Automatic Neural Network Configurations for the Classification of Complex Geographic Datasets, *Proceedings of the International Conference on Geocomputation,* University of Leeds, UK, pp. 343-358.

German, G. W. H., 1995, The Use of Multi Layered Perceptrons for Remote Sensing Classification with Temporal Data, *Proceedings of the International Conference on Neural Networks,* IEEE ICNN'95 Perth.

German, G. and Gahegan, M., 1996, Neural network architectures for the classification of temporal image sequences. To appear in: *Computers and Geosciences* (special edition on Neural Networks).

German, G., Gahegan, M. and West, G., 1997, Predictive Assessment of Neural Network Classifiers For Applications in GIS, *Proceedings of the 2nd International Conference on Geocomputation,* University of Otago, NZ.

Lees, B. G. and Ritman, K., 1991, Decision tree and rule induction approach to integration of remotely sensed and GIS data in mapping vegetation in disturbed or hilly environments. *Environmental Management*, Vol. 15, pp. 823-831.

Lees, B. G., 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.

Mardia, K. V., Kent, J. T. and Bibby, J.M., 1979, *Multivariate Analysis,* London Academic Press.

Rao, C. R., 1973, *Linear Statistical Inference and its Applications,* Wiley, New York.

Sarle, W., 1994, Neural Networks and Statistical Models, *Proceedings of the 19th Annual SAS Users Group International Conference,* SAS Institute, paper No. 320, pp. 1538-1550.

Wilkinson, G. G., Folving, S., Kanellopoulos, I., McCormick, N., Fullerton, K. and Megier, J., 1995, Forest Mapping from Multi-Source Satellite Data Using Neural Network Classifiers - An Experiment in Portugal, *Remote Sensing Reviews,* Vol. 12, pp. 83-106.