Predictive Assessment of Neural Network Classifiers For Applications In GIS

Gordon German, Mark Gahegan and Geoff West
Department of GIS, School Of Computing
Curtin University, P.O. Box U 1987, Perth 6001
Western Australia
email: gordon@cs.curtin.edu.au
Tel: +618 9266 3475, fax: +618 9266 2819

Presented at the second annual conference of GeoComputation '97 & SIRC '97, University of Otago, New Zealand, 26-29 August 1997

ABSTRACT

Artificial Neural Networks (ANNs) are well suited to implementing supervised classification tools for GIS data.They make no assumptions about the statistical nature of the data, can be used with ordinal and nominal data types together and can be trained with comparatively few training points, as they do not have to choose a data distribution model, unlike techniques such as Maximum Likelihood Classification.

However, training these neural network classifiers can be a time-consuming process, with no guarantee of the outcome. In this paper, the author presents a methodology for determining whether learning is practical for a given network on a given data-set, prior to commencement of the training phase.This is achieved by examining the error scores at the initial class boundaries and checking for redundancy in the network hyperplanes. This redundancy indicates how much flexibility is available in the network to learn complex boundaries.

1.0 Introduction

Neural networks have been used by the GIS community now for several years as an alternative tool for classification and feature extraction (Lees, 1994). Many commercial neural network software packages are currently used by the GIS professional and the next few years will likely see more of these classifiers integrated into GIS packages themselves.There now exists many variants of the original neural networks, as first proposed in the 1960s (eg.Amari, 1967), with a plethora of methodologies available to confuse the inexperienced user.This article is concerned with classification strategies and predicability of neural network classifiers, specifically in regard to our own neural net package, DONNET, which stands for Discrete Output Neural NET and is available via the World Wide Web on *http://www.curtin.cs/gis.*

Neural networks have often suffered from the dual spectres of difficulty-of-use and lengthy training times (Skidmore, 1995; Wray, 1996). In addition, the abundance of variations to the basic neural net paradigm available, makes it difficult for users not involved in the field to select the package most appropriate to their needs. DONNET is a neural network package specifically targeted towards users wishing to classify GIS data, across a range of statistical types (eg remote-sensed images, environmental surfaces, rock/soil classifications etc). In this paper, we examine how the learning phase associated with DONNET can be streamlined to the point where the GIS user can determine the suitability of the methodology to his/her particular task prior to actually starting the learning phase. The paper is organised as follows:

Section 2 describes DONNET's position in the hierarchy of artificial neural networks and gives a brief overview of the differences between DONNET and other networks. Although the final aim of the DONNET project is to produce a 'black box' classifier, it is often instructive to understand the broader machinations of such a tool. Section 3 is a refresher on the basics of neural network operations as they pertain to DONNET, briefly presenting the functional and conceptual models. Section 4 introduces methods of rating classification. We then introduce a methodology for assessing the learning capability of DONNET as a GIS classifier prior to training, using the data analysis abilities inherent in the network to highlight problem areas. An extension to the model is considered in Section 5. Finally, Section 6 presents the conclusions and the direction that further work will take in this field.

2.0 The Neural Network Hierarchy

2.1 Application Types

Neural networks cover essentially two broad automation tasks in Artificial Intellegence - function approximation and pattern recognition (Pao, 1989). In the former, the task is to learn a specific function of arbitrarily many dependent and independent variables from a set of known relationships and then interpolate for unknown dependent variables. In the latter, the task is to learn to recognise several distinct generalisations of specifically presented patterns and then pick these learnt 'classes' from a set of unknown patterns. In terms of actual architecture and implementation, these two types of nets can be quite similar, (if not identical), but the way in which their functional characteristics are modelled is quite different. This conceptual modelling is, of course, merely a tool to help in our understanding of the relationship between the network's processes and the task at hand, so should not be considered as a literal description of the network's implementation. DONNET is firmly geared towards the pattern recognition task, although with some modification, it can be implemented as a function approximator.

2.2 Classifiers

Within the domain of pattern recognition, or classification, there exists two main families of classifiers, *supervised* and *unsupervised*. This parallels the hierarchy within the statistical classification scheme, where such methodologies as K-means clustering and Maximum Likelihood Classification (MLC) provide examples of unsupervised and supervised classifiers respectively. An unsupervised classifier is free to pick its own generalised pattern structures (classes), i.e. it separates the given data into classes based on similarities it distinguishes within certain groups of examples of the data. As such, the user does not have any control on how to classify the data, or how many classes the data will be partitioned into. A supervised classifier is provided with a 'teacher', normally in the form of a file of target classes for each member of the training set. In this way, the user can control how the data is to be partitioned and, for this reason, supervised classification is generally more common for the classification of GIS data.

DONNET is a multi-layered perceptron (MLP) configured as a supervised classifier. It is customised for handling large, poorly separable datasets, with small sample sizes, corresponding to limited ground truth.

3.0 DONNET - An Overview

3.1 Introduction

Most of the material in this section has been covered in more detail in previous papers by the author (German, 1995; German & Gahegan, 1996) and others (Bischof et. al., 1992; Dunne, 1993). Here we simply present the basic concepts of DONNET, as viewed from both the functional and conceptual models.

3.2 Functional Overview of the MLP

The architecture of DONNET consists of an input layer, a hidden layer and an output layer. The input layer is passive, whereas the nodes of the hidden and output layers perform sigmoidal transformations of their input data. The number of input layer nodes (*p*) represents the dimensionality of the feature space, whilst the number of nodes in the output layer (*q*), is the number of partitions, or classes, we wish to impose upon this feature space. The number of hidden layer nodes (*h*), is dependent on *q* as:

$$h = (q \times (q-1))/2 \qquad q>2$$

the formulation and reasoning for this is presented in German & Gahegan (1996) and Gahegan et. al. (1996). Hence the architecture of the net is completely determined from the training data and the number of required classes, without assuming a particular distribution for the data.

Supervised classifiers learn by trying to minimise the error between the target set of outputs provided by the supervisor, and the learnt representation of the data by the classifier. DONNET's learning phase (as with most MLPs) can be analysed as four separate sub-phases:

1. For each sample (the input vector) in the training-set:
   a) Propagate the vector through the net.
   b) Compare the outputs with the target values and calculate an error figure.
   c) Back-propagate the error throughout the network's weight connections to give the *gradient* (derivative) of each weight with respect to the current error.
2. Update the weights using the derivative information, via some form of minimisation scheme to minimise the error figure.

The total error for the training set is then compared to some pre-set tolerance and, if not met, the whole training set is again fed through the net. This is termed one *epoch*, or *iteration*. It typically takes a few hundred epochs for DONNET to converge (reach a stable minimum error configuration).

3.3 DONNET Conceptual Model

Each hidden layer node, along with its associated weight connections to the input layer, represents a cutting hyperplane within feature-space. A particular hyperplane is moved through feature-space by changing the values of the weights going into the associated node. At start-up, DONNET creates enough hidden layer nodes to span all possible pairwise separations of the classes. It is the task of the learning phase to position and connect these hyperplanes so as to separate the classes in as effective a manner as possible. The starting position of these hyperplanes has been shown to dramatically affect the speed of convergence of the network to an optimal solution (see Dunne, 1992; German, 1994). The special instance of Fisher's Linear Discriminant for the two class problem (the first canonical variate) (Mardia et. al., 1974) is used to position each pairwise separating hyperplane, as a good approximation to the optimal discriminating position. In terms of the weight-space, we have fitted the network with weights such that the starting error is in the neighbourhood of an acceptable minima.

4.0 Predicting The Learning Ability of DONNET

4.1 Classification Accuracy

With real-world GIS classification problems, it is unrealistic to expect 100% classification accuracy on all data-sets. Therefore, once a classifier is trained, the user normally requires some indication of its accuracy, prior to being used on a particular data-set. Two types of accuracy are of interest to the GIS user:

- The ability of the net to learn from the data, i.e. its final ability to classify the training set.
- The ability of the net to generalise, i.e. it's performance on previously unseen data.

Accuracy is normally measured by passing a set of example data through the network and reporting on the number of correctly classified samples. Two points immediately arise from this:

- How to present the classification accuracy to the user.
- How to select an appropriate example set.

Selecting the appropriate example set is a delicate issue that is beyond the scope of this paper. Suffice to say that it is important to maintain statistical independence between the training set and example set used when the generalisability of the network is to be ascertained or measured (Sarle 1994). Here we are not concerned with the generalisability, but with the ability of a given network to learn from a particular data-set.

4.2 Presenting Classification Accuracy

The way classification accuracy is reported is affected by several factors:

- The relative number of samples in each class.
- The number of independent ground-truth sites available.

· The requirement of either overall, or class-by-class accuracy.

Generally, the reporting of neural network classifier accuracy has been a little inadequate. To some extent, this is due to most early neural network research work being done with artificial data-sets, in which the number of samples in each class is constant, as much ground truth as required is available and only *overall* accuracy is required. Subsequent researchers, in an attempt to directly compare their classifiers' accuracies with previous work, have tended to select data and report in a similar manner. Hence the most common form of reporting neural network accuracy is to give the total number of correctly classified samples in the example data-set, usually as a percentage (see Civco, 1993; Kamata, 1993 and Skidmore, 1995 for examples). This figure, (PCC, or percent correct) can be very misleading if the class sizes are not identical. As an exercise, compare Tables 1a and 1b, examples of a confusion matrix. Assume that classes 1 and 2 are very close in feature-space and hence more difficult to distinguish, whereas class 3 is easily separable (for example, lupins, wheat and water respectively). In Table 1a, all class sizes are equal, so the final figure of 43% PCC better reflects the difficulty the network has in separating classes 1 and 2 than in Table 1b (67% PCC), where the majority of samples are from class 3. A better approach is to quote the average per-class accuracy (the Average Normalised Response, ANR). This gives a figure of 43% ANR in both cases.

The ANR figure is useful for comparing the accuracy of different classifiers on the same data-set. However, for data

Table 1a : 3 class confusion matrix, equal class sizes

	Class1	Class2	Class3	Totals
True 1	**4**	2	4	10
True 2	3	**5**	2	10
True 3	5	1	**4**	10

Table 1b : 3 class confusion matrix, unequal class sizes

	Class1	Class2	Class3	Totals
True 1	**1**	3	1	5
True 2	0	**1**	4	5
True 3	1	1	**18**	20

analysis, one obvious problem with using simple single figures such as PCC and ANR is that they do not inform the user about the accuracy in any specific area of the data-set *feature-space*. Reporting each specific class accuracy or simply reproducing the confusion matrix conveys more information to the user on the classifier's performance, which can be used to help assess the confidence one has in the classification in different areas of the data-set.

4.4 Qualitative Analysis Of The Training Set Using The Network

We wish to *predict* whether a useful classification is achievable with a particular DONNET architecture, fitted with a set of starting weights, on a specific training set. Although this will not give us any indication of the generalisability of the network, it will allow us to decide whether or not the data set and the classification scheme we wish to impose are compatible with the DONNET methodology. We will satisfy ourselves, for the moment, with being able to answer the question:

Will training produce a significant improvement in the classification rate over our initial, or starting classification? To begin, we can make use of the initial weights used to identify difficult classification decisions. In the following, we assume that the costs of misclassification are identical for each class.

DONNET initially constructs the network with as many hidden-layer nodes (corresponding to separating hyperplanes in the feature-space) as needed to do a pairwise separation of every possible pairing of the classes. Hence, if there are four classes, six hyperplanes will be used, as there are six unique pairings of classes possible. In many cases however, one particular separating hyperplane may separate two or more other classes. For example, hyperplane A may be constructed to separate classes 1 and 2, but coincidentally may also separate classes 1 and 3, the task for which hyperplane B was constructed. In a sense, hyperplane B is "redundant" and could be used elsewhere if required, so moving the hyperplane out of this local region of feature-space will not increase the classification error between these three classes.

Pruning algorithms have been developed by both the decision tree and neural network communities for reducing the size of networks (see Brieman et. al., 1984, Dunne et. al., 1992). Dunne et. al. (1992) identifies each pairwise separation task occurring at every hidden-layer node by examination of the activated discriminant score given at the output of the hidden-layer nodes as the training set is passed through the network. This is done after training so that redundant nodes can be pruned from the network (usually a pruning tolerance is set to limit the level of pruning). Generally, these pruned networks do not perform as well as the unpruned network, since the trained network has already optimised its hyperplanes and hence most, if not all, are in use. However, the same algorithms can be used to identify redundant hyperplanes *prior* to training, some of which may not be used at all, or using the terminology of pruning methodologies, *exhibit zero confusion*. These redundant hyperplanes can be used by the network elsewhere in the feature-space, when needed. A task matrix can be constructed, similar to that in Table 2. The tasks listed in the Additional Separations column are tasks performed by that hyperplane as successfully as the hyperplane constructed to handle them as primary tasks. Hence, hyperplane A not only performs its own primary task of separating classes 1 and 2, but also separates classes 1 and 3 with the same or better rate of success as hyperplane B. Hyperplane B is therefore redundant in this localised region of feature-space. Note that this does not necessarily mean it should be pruned.

Consider the earlier three class problem, with 30 samples in the training set (see the confusion matrix of Table 1b). It is obvious from the confusion matrix that the classifier has correctly classified 18 out of 20 class 3 samples, but only 1 out of 5 for classes 1 and 2. We can determine

Table 2: Three Class separation. Here, hyperplane B is redundant

Hyperplane	Primary Class Separation	Additional Separations
A	1:2	1:3
B	1:3	-
C	2:3	-

more than this from the matrix, however. The rows are the true class allocations and give the errors-of-omission, or conditional probability distributions. The columns give us the errors-of-commission. So from Table 1b, the classifier has not only classified 18 samples of class 3 as class 3, but also 4 samples from class 2 and 1 from class 1 (reading up column 3). In fact, totaling the symmetric off-diagonal positions will produce figures for the total errors of classification (the misclassification rate) for each pairwise separating hyperplane. Table 3 gives this "boundary misclassification rate" (BMR) as a block diagonal matrix for each of the three separating hyperplanes. It can be automatically derived from the confusion matrix of Table 1b. The AVG column gives the average misclassification for all boundaries associated with the respective class; hence the average BMR of 19% for class 2 is calculated from the summation of all class 2 columns and rows. We can see that the class 1:2 hyperplane is the largest contributor to the ANR classification error and in general, the average figure of 25% shows that the class 1 region of feature-space delineated by the hyperplanes is contributing to the most error.

Table 3 : Boundary misclassification rate as a percentage of class sizes

	Class1	Class2	Class3	**AVG**
Class1	-	+30%	**20%**	**25%**
Class2	-	-	**8%**	**19%**
Class3	-	-	-	**14%**

Some entries in the BMR matrix have a positive or negative sign associated with them. The positive sign indicates that all the errors are errors of omission, the negative signs indicate that all are errors of commission. Non-zero figures that comprise of both (dual-error boundaries) have no sign and are highlighted in bold. These differences are important, as they indicate whether or not a simple movement of the associated hyperplane will reduce the local misclassification or not. Figure 1 shows the differences graphically for the two dimensional case, with the separating hyperplane shown as a heavy black line. In Figure 1a, a single-error boundary is shown for two non-overlapping classes (in feature-space). A simple local repositioning of the separating hyperplane will reduce this error, ideally to zero, by positioning it in the region between the classes. Figure 1b shows the case where there is overlap of the classes and a single-error boundary has been formed. In this case, the error can be quickly minimised, again by a local repositioning of the hyperplane, which will reduce it to a dual-error boundary. However, if the hyperplane is producing a dual-error figure, as in Figure 1c, then simple movement of the hyperplane will not suffice to reduce the error significantly and the boundary complexity will have to be increased. This can only be accomplished by either varying the summation of the regions (accomplished at the output layer level of the network) alone, or by additionally moving across one or more redundant hyperplanes, to aid in building up the complexity of the surface. The task matrix can identify whether or not there are redundant hyperplanes available.

The BMR matrix can therefore be used to identify problem boundaries and determine if a simple movement of the hyperplane is sufficient. Along with the task matrix, we can also determine whether or not there is sufficient redundancy in the number of hyperplanes to reduce the dual-error boundaries' misclassification. Ideally, we would expect the following behaviour during training, in terms of the BMR matrix:

1. A reduction of single-error boundary figures to zero, or else a transformation to a lower-valued dual-error figure.
2. Dual-error figures to be reduced only if there are sufficient redundant hyperplanes available, or the addition/subtraction functions of the output layer of the network can be further optimised, to increase the complexity of the boundary.
3. Zero error boundaries to remain in the same state.
4. We would not expect a transformation of dual-error to single-error boundaries, as this would not generally reduce the local misclassification rate (see Figure 1c).

The ability of the network to reduce its classification error can therefore be determined from these matrices. A localised repositioning of the separating hyperplane can reduce single-error figures in the BMR matrix. If there are any redundant hyperplanes in one area of the feature-space, they can be shifted elsewhere to be used to model more complex boundaries and so reduce some of the dual-error figures.

4.6 An Example

Table 4 shows a typical confusion matrix for a DONNET classifier immediately prior to training. The data-set is 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. Table 5 is the associated BMR matrix. The classes are of unequal sample sizes, with the class totals in column 10. The initial classification rate (before training) is 52.79% ANR (72.15% PCC). Glancing down the main diagonal of

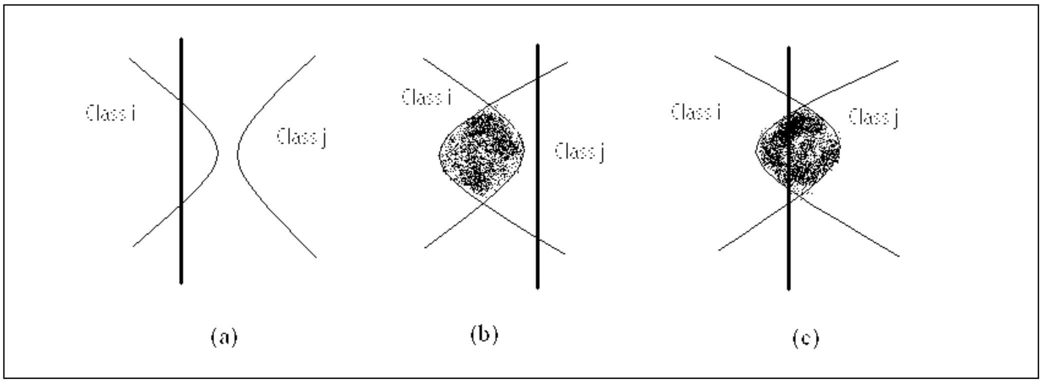

Figure 1 : (a) single error boundary, reducible to zero. (b) single error boundary, reducible to dual. (c) dual error boundary.

Table 4, we see that the initial network configuration labels classes 1, 4, 5, 7 and 8 quite well. However, there are many errors of omission for class 1, as indicated by its column entries. The conclusion one can draw from this is that the class 1 region delineated in feature-space by the bounding hyperplanes is too large - the hyperplane fit is too loose. In contrast, considering the class 3 column, the hyperplane fit is too tight - all but one example have been excluded from the class.

Analysing the BMR matrix of Table 5, we find that the class boundaries 1:4, 4:5 and 4:6 overlap considerably. Only 11 of the 36 boundaries are dual-error boundaries, there are 12 single-error boundaries and 13 boundaries with no error (overlap) at all. The class 9 feature-space region is correctly delineated; in fact, it is likely that many of the associated hyperplanes are redundant and can be used elsewhere. This applies, to a slightly lesser extent, to class 8 as well and is confirmed by the task matrix, a portion of which is shown in Table 6. There are, in fact, 16 redundant hyperplanes at the commencement of training ie. 16 hyperplanes whose primary separation task is performed with the same success by other hyperplanes.

We can conclude, from this rudimentary assessment of the initial task and BMR matrices, that training will give us a significant improvement in classification due to the following:

1. There are a significant number of single-error boundaries, hence fine-tuning of the associated hyperplane positions should improve these.
2. There is sufficient redundancy in the number of hyperplanes to aid in the construction of more complex boundaries for some of the 11 cases where we have a dual-error boundary.

Table 4 : 9 Class Confusion Matrix (0 iterations)

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Totals
True 1	**187**	3	0	13	17	3	0	7	0	230
True 2	26	**8**	0	5	2	1	3	5	0	50
True 3	26	2	**1**	0	4	1	3	2	0	39
True 4	39	1	0	**108**	30	0	11	0	0	189
True 5	22	0	0	28	**85**	0	1	0	0	136
True 6	19	0	0	31	2	**18**	3	0	0	73
True 7	17	2	0	14	5	6	**22**	0	0	66
True 8	3	0	0	0	0	0	0	**122**	0	125
True 9	0	0	0	0	0	0	0	0	**374**	374

Table 5 : 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 6 : Portions of Task Matrix for 9 Class Example

Hyperplane	Primary Class Separation	Additional Separations
1	1:2	-
2	1:3	1:9, 2:9, 3:7, 3:9, 4:6, 5:7
28	5:7	-
29	5:8	4:8, 6:8
30	5:9	1:6
36	8:9	1:8

In fact, when trained on this data-set for 250 iterations, the network produces the confusion and BMR matrices of Table 7 and Table 8. The classification rate is now 65.66% ANR (with 78.55% PCC). As expected, most single-error boundaries have been reduced to lower valued dual-error boundaries or zero (12 down to 6). Dual error figures have been reduced for all class 1 boundaries, as have all class 4 figures (these were the two classes contributing most to the overall error rate), indicating that boundary complexity has been *increased* around these classes. Again, as expected, no dual-error boundaries have been transformed into single-error boundaries and all zero-error boundaries have remained the same. The overall local misclassification rates associated with class 2 have increased slightly (errors of commission have increased). Class 3 errors are approximately the same and all others have decreased slightly. There are still some single-error boundaries left, so we could expect marginal improvements on the classification rate with further training (but probably at the cost of reduced generalisability).

The procedure for examination prior to training and the information that can be derived is thus as follows:

1. Fit the network with the weights calculated from the discriminant functions.
2. Generate confusion, BMR and task matrices.
3. If there are a significant number of single-error figures

Table 7 : 9 Class Confusion Matrix (250 iterations)

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Totals
True 1	**195**	4	2	9	15	3	1	1	0	230
True 2	10	**26**	0	3	2	1	4	4	0	50
True 3	18	5	**7**	0	4	0	3	2	0	39
True 4	38	1	0	**120**	15	4	11	0	0	189
True 5	18	3	0	27	**84**	1	3	0	0	136
True 6	7	2	1	24	3	**34**	2	0	0	73
True 7	7	2	0	8	3	3	**43**	0	0	66
True 8	1	0	0	0	0	0	0	**124**	0	125
True 9	0	0	0	0	0	0	0	0	**374**	374

Table 8 : 9 Class BMR Matrix (250 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%**

in the BMR matrix contributing to the average BMR figures, a worthwhile reduction in error can be expected.
4. If there are redundant hyperplanes available, reductions in dual-error figures can be expected.
5. If all error-figures are dual-error and all hyperplanes are in use, no further significant error reduction can be expected.

5.0 Extending the Model

There are two points worth noting about the conceptual model:

- By using the pairwise linear discriminants to calculate the number of hyperplanes and the corresponding initial weights, the net starts from the assumption that the data is linearly separable. This is not as big a restriction as it first seems, if one has at least several output classes. Unless all class centroids are closely clustered within a particular class distribution in feature-space (if this is the case, it is unlikely that any classification strategy will work), at least some of them will be linearly separable. Furthermore, as we have shown, although each hyperplane is linear, complex boundaries can be built up from a combination of hyperplanes, allowing some of the non-linear boundaries to be modelled. The weights between the hidden and output layers allow addition and subtraction of isolated clusters of data, which helps the network classify non-unimodal distributions, as well as aiding in the construction of more complex surfaces from individual hyperplanes.
- As implied above, the complexity of the decision boundaries is directly related to the number of output classes we designate.

The last point is worth further discussion. DONNET may fail to classify a high dimensional feature-space into just two classes, for instance, because of a lack of complexity in the discriminating boundary (in this case, there would be just one hyperplane). Additionally, the situation can arise where all hyperplanes are in use and some of the discriminating boundaries still require a higher level of complexity. These situations can be overcome by the addition of more hidden layer nodes, corresponding to adding more hyperplanes. The problem arises as to how to position these new hyperplanes so as not to disturb the network's position in weight-space too dramatically. The addition of new random weights and nodes to a network will radically shift its position in weight-space, normally with a significant increase in error. If we wish to keep our conceptual model, a simple solution is to re-classify the classes on either side of the hyperplane in question into artificially conceived sub-classes, which increases the complexity available for the decision boundaries. Then we can calculate the discriminant functions for the new weights and continue training, without disturbing the current position of the network in weight-space significantly. We can finally do a sub-class merge to return to the original number of classes.

6.0 Conclusions

The BMR matrix can be used to identify successful discriminating boundaries and those that may require additional hyperplanes. The task matrix can identify any redundant hyperplanes in the system. Production of these matrices can be automated and implemented at any stage of the training phase. Examination of the BMR and task matrices prior to the commencement of training, after fitting the model with the weights derived from the discriminant functions, can show the user where likely improvements can be made to the classification. This examination process can also be automated and can quickly signal the user if there is not enough flexibility in the model to improve the classification rate beyond that produced by the initial conditions. This only takes one pass through the training-set, as compared to the several hundred needed to train the network.

Further work is under-way to automate the spawning of additional nodes when required, as discussed in Section 5.0. Alternatively, the user may wish to stop when the hyperplane redundancy in the network has been depleted. The task matrix can be analysed during training to indicate when this is so. A useful metric for reporting on the spatial distribution of the error, using the error map generated by DONNET should also be developed. This will depend, to a large extent, on the spatial distribution of the training-set and whether the training sites are isolated points or homogeneous regions across the data-set.

References

Amari S. (1967) Theory of Adaptive Pattern Classifiers, *IEEE Transactions on Elect. Comput.*, Vol. EC-16, pp. 299-307

Bischof H., Schneider W. and Pinz A.J. (1992) Multispectral classification of Landsat images using neural networks, *IEEE transactions on Geoscience and Remote Sensing*, Vol. 30, No. 3, pp. 482-490

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

Civco D. L. (1993) Artificial neural networks for landcover classification and mapping, *International Journal of Geographical Information Sytems,* Vol. 7, No. 2, pp. 173-186

Dunne R.A., Campbell R.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.A., Campbell R.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, *Proc. Int. Conf. on Geocomputation*, University of Leeds, UK, pp. 343-358.

German G.W. H. (1994) Multi Layered Perceptrons and the Classification of Remotely Sensed Data, *Honours Thesis,* School Of Computing, Curtin University of Technology Western Australia

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.W.H. and Gahegan M. N. (1996) Neural Network Architectures For The Classification Of Temporal Image Sequences, *Computers And Geosciences,* Vol. 22, No. 9, pp. 969-979

Kamata S. and Kawaguchi E. (1993) A neural net classifier for multi-temporal {LANDSAT} images using spatial and spectral information, *Proceedings of 1993 International Joint Conference on Neural Networks,* IEEE , Vol. 3, pp. 2199-2202

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, *7th Australasian Remote Sensing Conference Proceedings,* pp. 51-60

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

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

Sarle W. (1994) Neural networks and statistical models, *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, SAS Institute, paper No. 320, pp. 1538-1550

Skidmore A. (1995) Neural Networks and GIS, Australasian Geographic Information Systems Applications, No. 11, pp. 53

Wray B. (1996) A guide to the literature on learning probabilistic networks from data, *IEEE Transactions on Knowledge and Data Engineering,* Vol. 8, No.3 (in press)