The identification of sub-pixel components from remotely sensed data: an evaluation of an artificial neural network approach

Alice Clara Bernard

THE IDENTIFICATION OF SUB-PIXEL COMPONENTS FROM REMOTELY SENSED DATA: AN EVALUATION OF AN ARTIFICIAL NEURAL NETWORK APPROACH

Alice Clara Bernard

Thesis submitted for the degree of Doctor of Philosophy

1998

Abstract

Until recently, methodologies to extract sub-pixel information from remotely sensed data have focused on linear un-mixing models and so called fuzzy classifiers. Recent research has suggested that neural networks have the potential for providing sub-pixel information. Neural networks offer an attractive alternative as they are non-parametric, they are not restricted to any number of classes, they do not assume that the spectral signatures of pixel components mix linearly and they do not necessarily have to be trained with pure pixels. The thesis tests the validity of neural networks for extracting sub-pixel information using a combination of qualitative and quantitative analysis tools.

Previously published experiments use data sets that are often limited in terms of numbers of pixels and numbers of classes. The data sets used in the thesis reflect the complexity of the landscape. Preparation for the experiments is carried out by analysing the data sets and establishing that the network is not sensitive to particular choices of parameters.

Classification results using a conventional type of target with which to train the network show that the response of the network to mixed pixels is different from the response of the network to pure pixels. Different target types are then tested. Although targets which provide detailed compositional information produce higher accuracies of classification for subsidiary classes, there is a trade off between the added information and added complexity which can decrease classification accuracy. Overall, the results show that the network seems to be able to identify the classes that are present within pixels but not their proportions. Experiments with a very accurate data set show that the network behaves like a pattern matching algorithm and requires examples of mixed pixels in the training data set in order to estimate pixel compositions for unseen pixels. The network does not function like an unmixing model and cannot interpolate between pure classes.

THE IDENTIFICATION OF SUB-PIXEL COMPONENTS FROM REMOTELY SENSED DATA: AN EVALUATION OF AN ARTIFICIAL NEURAL NETWORK APPROACH

Alice Clara Bernard

Thesis submitted for the degree of Doctor of Philosophy

University of Durham

Department of Geography

1998

Table of Contents

List of Figures

Chapter V:

Chapter VI:

Chapter VII:

Chapter VIII:

List of Tables

Chapter IV:

Chapter V:

Chapter VI:

Chapter VII:

Chapter VIII:

List of Acronyms

AVHRR	-Advanced Very High Resolution Radiometer
DN	- Digital Number
DBH	- Diameter at Breast Height
EMAP	- Environmental Mapping and Modelling Unit
GIS	- Geographical Information System
GPS	- Global Positioning System
JRC	- Joint Research Centre
NDVI	- Normalised Difference Vegetation Index
MLP	- Multi-Layer Perceptron
rmse	- root mean square error
SPOT	- Système Probatoire d'Observation de la Terre
TM	- Thematic Mapper

Declaration

The work contained in this thesis is entirely that of the author. Material from the published work of others, which is referred to in this thesis, is credited to the author(s) in question in the text. No part of this work has been submitted for any other degree in this or any other university.

The main body of text (excluding references and appendices) is approximately 70,000 words in length and conforms with the word limit.

Alice C. Bernard

Copyright

Acknowledgments

This thesis is dedicated to my parents to whom I once swore
"I will never, ever, do a PhD".

It is customary in this section to thank a number of people who have helped during the research and write up of the PhD thesis, particularly supervisors, system managers, friends, family, and so on. One person, however, never gets thanked even though present at every stage. Well, I am going to break with tradition and thank the person who had to do all of these: come up with ideas, research the subject, carry out experiments, develop tools, analyse results, draw conclusions and, most importantly, write the thesis up - the author!

That said, many, many people did help me along the way, and I would like to sincerely thank them all. It may have been help with computing problems - 'No, you cannot have 10 Gigabytes of disk space!', data problems - 'How can 60% + 45% make 100%?', ideas - 'If you talked to your program, maybe it would work better' or suggestions - 'If I were you, I would give up'. Or it may simply have been regular 5mn coffee breaks, two words of encouragement, the worst joke ever, a daily email or a sympathetic ear. The friend who offered to fly 2000 miles to help me do some photocopying deserves a special mention.

Most of all, I would like to thank those who had to endure me while I finished writing the thesis up, particularly those who lived to regret the famous words 'of course you can come and stay, for as long as you like'. I am very grateful to my supervisor who read countless iterations of each chapter.

I am thankful to my colleagues at the Joint Research Centre of Ispra, Italy, where I was based, for many happy years spent in their company and to the European Commission, for providing me with a grant under their 'Training and Mobility' program.

Finally, thanks go to my very best friend, for making sure I ate properly, for forgiving my foul moods, for looking after my cat, and for waiting 'just another week' for me to finish.

Chapter I

INTRODUCTION

Remotely sensed satellite images now routinely complement mapping techniques but there has been increasing concern about using the pixel as the fundamental unit of classification without due concern for its shortcomings (Fisher, 1997; Cracknell, 1998). Digital images divide up the landscape into arbitrary units, pixels, to which conventional classifiers assign one landcover category. However, the nature of landcover is such that one class often does not adequately represent the area that a pixel covers.

Neural networks have become well established over the last decade as classifiers of remotely sensed data, as the recent special issue of the *International Journal of Remote Sensing* devoted to neural networks (Vol. 18, N° 4) testifies to. Recently, empirical results have suggested a strong potential for a relationship between neural network output values and sub-pixel information (Moody *et al.*, 1996; Atkinson *et al.*, 1997; Foody *et al.* 1997; Warner and Shank, 1997). The research carried out in this thesis seeks to provide a better understanding of the relationship between neural network output values and sub-pixel information.

1.1 The Problem with Mixed Pixels

Producing a satellite image can be likened to placing a grid over the earth's surface and recording, through an electric signal, the average intensity of reflected and emitted radiance from the area covered by each grid cell, or pixel, for the entire scene. Each channel records radiation in a range of wavelengths and the analog (continuous) signal is sampled at regular intervals; the values of the samples are averaged to produce a digital number (Lillesand and Kiefer, 1994).

Classification is the process of grouping objects which share similar characteristics under the same label and differentiating them from objects grouped under a different label. It can only be completely successful if at least one characteristic of objects from different groups is unique to each group. Theoretically, every different landcover type is represented by a unique spectral signature. The purpose of image classifiers is to produce

a landcover map by separating pixels into different categories, or classes, based on the signature of the landcover of the area covered by the pixel, henceforth called the pixel's signature.

The often unstated assumption and requirement of conventional classification methodology is that pixels can and must always be included or excluded as a whole within one category (Richards, 1986; Schott, 1997). However, at any resolution, some pixels will contain more than one category, a direct consequence of the nature of the landscape and of the spatial resolution of the sensor. These pixels are called mixed pixels and may contain several distinct categories, several intimately mixed categories, or combinations of both types. Mixed pixels with distinct categories include those which lie on well defined borders between two or more classes and those with sub-pixel features. Mixed pixels with intimately mixed categories include those with composite landcover classes such as *Vineyards* or *Orchards*, which are homogeneous mixtures of simpler classes such as *Vines* or *Fruit Trees,* and *Bare Soil,* and those with vertically mixed categories, such as in a forest, trees and undergrowth. Mixed pixels that are combinations of different mixing types include those lying on the border between landcover types that merge one into the other. Examples of mixed pixels are illustrated in figure 1-1.

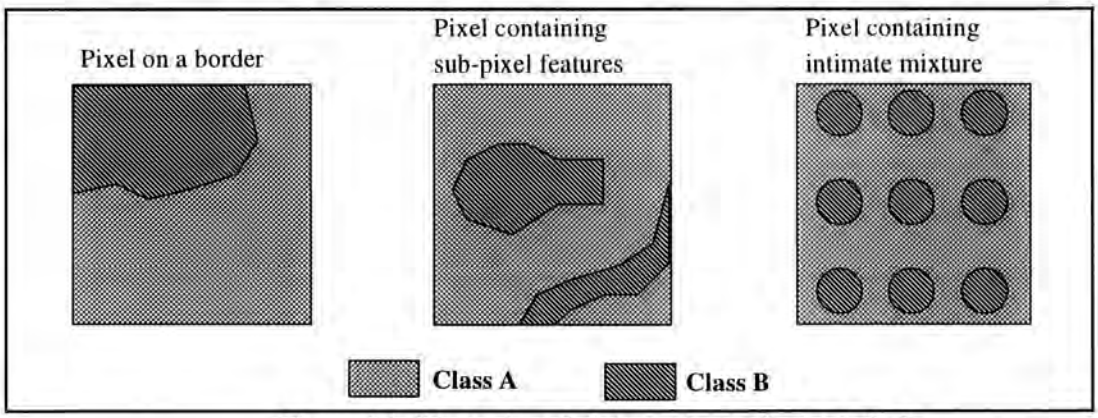


Figure 1-1. Illustration of different types of mixed pixels

The problem with mixed pixels is that their spectral signatures are combinations of the spectral signatures of their components and therefore differ from the typical spectral signatures of any of their components. The process by which the sub-pixel signals combine is not well understood. Physical theory suggests that signals may combine linearly and the simplest mixing models make this assumption. But as landcover types increase in number and mixing becomes more intimate, the close association of classes becomes increasingly non-linear (Borel and Gerstl, 1994; Ray and Murray, 1996). The presence of pixels which do not resemble the training clusters, or worse, whose signal resembles that of a class not present in the pixel, is a source of confusion for the classifier

and increases misclassification (Chhikara, 1984; Campbell, 1996; Foschi and Smith, 1997).

The exact extent of the effect of mixed pixels on classification is not well known since the methodology for many projects is to specifically remove mixed, or potentially mixed, pixels from the training and testing sets (Arai, 1992; Yoshima and Omatu, 1994). Ignoring the presence of mixed pixels is not a solution since mixed pixels will exist in the image, particularly for low resolution sensors, and therefore although the testing accuracy may well be high, it will not be representative of the accuracy of classification of the image.

Simply increasing the resolution of the satellite imagery may not be practical nor does it guarantee less mixed pixels since new features may resolve. Additionally, since lower resolution classes can be more homogeneous, as spatial resolution increases so may spectral variability and consequently separability may decrease (Irons *et al.*, 1985); increased resolution does not necessarily imply increased classification accuracy (Fisher and Pathirana, 1990).

The presence of mixed pixels may decrease classification accuracy but more importantly, the information present in mixed pixels may be valuable and assigning only one landcover type may not be appropriate. For example, area estimates would be more accurate if class proportions, that is the proportion that individual classes occupy in a pixel, could be estimated. In some cases, a secondary class may be of more interest than the dominant class. For example, a forest inventory may require knowledge that trees exist within the area covered by a pixel, even if they do not cover the majority of the pixel.

1.2 Objectives of the Thesis

In conventional classification methods, even when a category represents a mixture, the pixel as a whole belongs to that category: for example 'mixed oak/pine/weeds' (Wilkinson *et al.*, 1995) or a category which is used for areas of between 25% and 75% pine cover (Brockhaus and Khorram, 1992). This premise defines what is referred to as 'pure' classification, also sometimes called 'hard' or 'crisp' classification. Henceforth, the term 'pure' will be italicised to emphasize the possibility that the pixel may not actually be *pure* but is treated as such. The existence of mixed pixels highlights the need for a methodology which can not only deal with them but also estimate their composition. Classification methods that seek to provide information about the composition of a pixel will be referred to as *soft* classifiers in this thesis, in contrast to *pure* classifiers.

Recently, it has been suggested that output values from a neural network, specifically a multi-layer perceptron network, may be correlated with class proportions, also called percentage cover, within pixels (Moody *et al.*, 1996; Atkinson *et al.*, 1997; Foody *et al.*, 1997; Warner and Shank, 1997). Neural networks have a number of features which make them attractive for this task. In particular, the data need not conform to parametric statistical requirements, and signals from sub-pixel components are not assumed to mix linearly. The overall aim of the thesis is to **investigate the relationship between neural network output values and sub-pixel information**.

In so doing, a number of methodological issues will be raised that the majority of experiments reported in the literature and discussed in chapter II have not addressed. The consequences of choosing particular parameters for the network used in the thesis (henceforth simply referred to as 'the' neural network), for example architecture, number of iterations, learning rate, and so on, are examined. The reference data and the data sets created are described and analysed in detail so that any possible influence on results may be known. Data sets are created which appropriately reflect the nature of the landscape and the complexity of the classification problem. The effect of different types of target representations of ground data is examined. Finally, the method of representation of mixed pixels by the network is explored. In summary, part of the aim of this thesis is to **address methodological issues concerning the identification of sub-pixel information by the neural network**.

The literature also shows a lack of standard tools for the analysis of *soft* classification results. The majority of techniques analyse the relationship between ground cover proportions and classifier output, class by class, or else provide a single overall accuracy value. Using these methods, the relationship between components in a pixel is lost and the distribution of results is not available. Qualitative, visual, and quantitative, numeric, analysis tools are developed and used to evaluate the performance of networks in identifying sub-pixel components. Within the thesis, therefore, another aim is to **develop tools for the analysis of *soft* classification results**.

1.3 Structure of the Thesis

The thesis starts with an overview of relevant work published in the literature: **Chapter II**: *Prior Research in Identifying Sub-Pixel Information.* This chapter considers the research that has been carried out to-date in identifying sub-pixel components and their proportions. Methodological issues and results form the basis for the approach chosen in this thesis.

At the European Commission's Joint Research Centre of Ispra (Italy) at which the author was based, a network had already been developed for the purpose of *pure* classification of remotely sensed images. **Chapter III**: *The Multi-Layer Perceptron Network* describes the characteristics of this network and the issues that must be considered when running it. An outline of the modifications carried out to the network so that it could perform *soft* classification is also provided. This chapter is complemented by **Appendix A**: *Soft Classification Software Package.*

Chapter IV: *Reference Data Sites and Data Sets* provides a detailed description of the data sets and the methods for creating them. Details of the satellite imagery, photographs of the landscape, descriptions of the ground campaigns, explanations concerning the creation of the data sets and analysis of the data sets are included. The chapter is complemented by **Appendix B**: *Reference Data Information.*

Having introduced the methodological tool and the data sets, the experiments start with **Chapter V**: *Sensitivity Analysis of the Network* which reports on tests carried out to confirm that particular choices of network parameters would not affect classification output values. All the parameters of the network are tested in turn and conclusions as to their effect on overall accuracy are drawn.

Chapter VI: *Qualitative Analysis Tools and Sub-Pixel Information* groups experiments which use established and newly developed visual tools to study the relationship between pixel components and neural network output values. Analysis is qualitative in that it is performed visually, not numerically. The chapter describes the analysis tools then uses them to evaluate the correspondence between neural network output values and pixel components.

The issue of training strategy is addressed in **Chapter VII**: *Quantitative Analysis Tools and Target Types* which investigates the effect of different target types to represent reference data on *soft* classification accuracies. Two new types of matrices are developed to provide quantitative analysis tools for calculating and comparing classification accuracies. The matrices are used to determine how target types affect the neural network results.

Chapter VIII: *Composition of Training Data Sets* is devoted to establishing whether the neural network is capable of interpolating between values to provide the composition of a mixed pixel or whether it requires examples of mixed pixels in the training file in order to approximate compositions of pixels in the testing file.

The last chapter of the thesis, **Chapter IX**: *Summary and Conclusions,* provides a

synthesis of the experiments that have been carried out and highlights their results and the qualitative and quantitative analysis tools that have been developed. General conclusions regarding the appropriateness of a neural network for extracting sub-pixel information are drawn. Aspects of the subject which have not been fully addressed are suggested for future work.

Chapter II

PRIOR RESEARCH IN IDENTIFYING SUB-PIXEL INFORMATION

As was highlighted in chapter I, mixed pixels occur within any raster image based remote sensing system, whatever its resolution. Mixed pixels may decrease classification accuracy because their spectral signature is likely to be different from the mean spectral signature of *pure* classes. Therefore, mixed pixels cannot simply be ignored; removing mixed pixels from the training and testing data sets might produce high classification accuracies of the testing data but that will not be representative of the accuracy of classification of the image. Furthermore, the composition of pixels is often of interest. Classification methods which provide sub-pixel composition are therefore very useful.

A number of methods have been developed to identify sub-pixel classes and their proportions. Simple vegetation indices such as the NDVI (Campbell, 1996), which has been shown to be strongly correlated with leaf area index, could be used as a crude measure of vegetation proportions but they are only appropriate for two component mixtures. Complex non-linear mixing algorithms have been developed which postulate a model for the relationship between class proportions and the spectral signature of a pixel by modelling spectral behaviour (Borel and Gerstl, 1994; Jasinski, 1996; Li and Strahler, 1986). However, these mixture models assume that the user has detailed knowledge of the complex physical behaviour of the interaction between sunlight and objects. This overview of research to-date will concentrate on the most common algorithms in the remote sensing literature. These can be divided into three groups.

- Algorithms which postulate a linear relationship between category proportions in a pixel and the spectral signature of the pixel, usually called linear unmixing models.
- Algorithms which are based on some form of distance measurement of the pixel from the spectral mean of categories and which produce likelihood or membership values. These include modified maximum likelihood and so-called fuzzy techniques.

- Algorithms which extract the relationship between spectral signatures and ground cover proportions through examples using parallel like processing. These are termed neural networks.

For each of these algorithms, the basic methodology is explained and results and limitations are discussed.

2.1 First Considerations

There is an important issue which needs to be clarified. Pixel signatures are not only a result of the categories that compose them. Conditions of identical landcover types may vary. For example, they may be found at different physical heights, under different illuminations, with a different water content, and so on, all of which will affect the signal received by the sensor. Technological considerations and atmospheric scattering and absorption limit the wavelengths at which radiances can be measured and the sensor only records approximate spectral signatures dictated by its spatial resolution, the area covered by one pixel; its radiometric resolution, that is the number of brightness levels which can be distinguished; and is spectral resolution, the width of wavelength intervals (Richards, 1986). Signals from neighbouring pixels may also contaminate a pixel's reflectance curve (Campbell, 1981; Cracknell, 1998). Therefore, pixels containing identical classes are likely to have similar but not identical signatures and different landcover classes may give rise to spectral signatures which are similar to one another.

Landcover classes are represented by clusters of pixels, as illustrated in figure 2-1, and classification theory suggests that the further from the centre of a cluster a pixel is found, the more unlikely it is to belong to that cluster. However, the reasons for a pixel being on the edge of a cluster, such as pixels **A, B** and **C** may differ. The signal from pixel **A** might be the result of a mixture between landcover classes $\mathbf{L_1}$ and $\mathbf{L_2}$ since it lies between the two centroids. However pixel **B**, situated at a similar distance from the

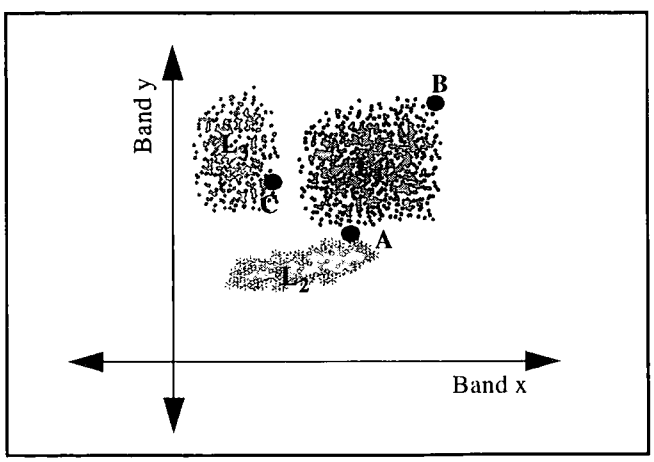


Figure 2-1. Illustration of category clusters and mixed pixels

centroid of class L_1 as **A**, but not close to any other classes, would probably not be considered a mixed pixel. Instead, the signal from pixel **B** might be considered to be a result of conditions on the ground and, although it may be considered to have a low level of similarity to the mean spectral signature of landcover class L_1, it would be considered to belong in full to L_1. Pixel **C** could be in a similar situation to pixel **B**, actually covered by class L_3, or it could be a non-linear mixing between classes L_1 and L_2 or it could contain all three landcover classes, L_1, L_2 and L_3.

There appears to be no method for differentiating between these types of pixels. For this reason, it is essential that accurate reference data be collected. Only when the actual composition of a pixel is known can classification results be evaluated. This is also the reason why supervised classification is to be preferred over unsupervised classification methods. Even for *pure* classification, it is sometimes difficult to assign landcover classes to the clusters that unsupervised methods have identified. With *soft* classification, the task becomes even harder.

2.2 Linear Unmixing Models

Linear mixture models have been successfully used in many applications at a range of resolutions. Examples include the decomposition of minerals from Mars using imagery from the Viking Lander 1 (Adams *et al.*, 1986), routine mapping of intertidal vegetation in an estuary using Landsat TM data (Reid-Thomas *et al.*, 1995) and crop (Quarmby *et al.*, 1992) and tropical forest (Cross *et al.*, 1991) monitoring with AVHRR imagery.

2.2.1 Methodology

The linear mixture model assumes that the observed spectra from a pixel is given by the product of the reflectivity of a category and the fraction of the pixel that category occupies. A series of assumptions are made that include a source of even illumination of known spectrum and intensity and no fluorescence and photons only interacting once with the surface (Horwitz *et al.*, 1971; Detchmendy and Pace, 1972; Settle and Drake, 1993). This model is usually extended to include an error term which includes instrument, atmospheric and other contributions to the observed signal (Hallum, 1972). The model can be expressed by the following equation.

$$E = \sum_{i=1}^{N} x_i \cdot f_i + e \qquad (2.1)$$

where $\boldsymbol{E}$ is the observation, that is the spectral signature from the pixel; $\boldsymbol{x}_i$ is the *pure* signal from component $\boldsymbol{i}$, usually referred to as an end-member and taken to be the

response that would be received by a pixel containing only category i; f_i is the fraction of the pixel that component i covers and e_i is an error term. Since this equation represents a physical reality, f_i is usually constrained to satisfying the following equations:

$$0 \leq f_i \leq 1$$
$$\sum f_i = 1 \qquad (2.2)$$

In other words, individual proportions must sum to one and cannot have values less than zero or greater than one. In most methods, the error term is assumed to have statistical properties that are independent of the mixture and that can be represented by a normal distribution of mean zero determined by the covariance matrix of the end-members which is constant and equal to C (Settle and Campbell, 1998).

The linear mixture model can be solved if:

- spectral signatures of *pure* components, x_i, are known and thus fractions of the pixel occupied by each component, f_i, can be calculated, or
- if fractions of the pixel occupied by each component, f_i, are known and thus *pure* spectral signatures, x_i, can be estimated.

End-member signals, $\mathbf{x_i}$, are either known as part of a library of laboratory measurements or estimated from the image. Proportions of components within pixel, $\mathbf{f_i}$, are either known from ground data or can be estimated from imagery with a higher resolution than that which is being analysed. In either of these cases, the model is solved by multi-variate linear regression or some form of least squares analysis, where the sum of squares of the error is minimised (Shimabukuro and Smith, 1991; Settle and Drake, 1993)

2.2.2 End-members and limitations

One of the particularities of the linear mixture models is that they often include a shade component as one of the end-members. Shadow effects depend primarily on sun angle, ground morphology and, if present, the type and structure of vegetation within the image (Jupp and Walker, 1997). Shadows can be an important factor (Horler and Ahern, 1986) which may cause landcover reflectances to be underestimated (Asner *et al.*, 1997) but not all authors report an influence (Stenback and Congalton, 1990).

In some cases, examples of the *pure* spectra of objects in an image may be available in a library of laboratory spectra which have been measured under rigorously controlled conditions and known illumination conditions. This is the method used for example by Adams *et al.* (1986) to determine the composition of objects in an image taken of the surface of Mars using a sensor on-board the Viking Lander 1. Alternatively, a library of

spectra can be created *in situ* for the image by approximating laboratory conditions and taking spectrometric measurements of the objects during a field campaign. Sohn and McCoy (1997) use field measured spectra to map arid rangeland. However, the observed image spectra are not the true reflectivity from the ground elements since the reflected signals pass through the atmosphere before reaching the sensor. Therefore, the spectra obtained from the image need to be corrected for atmospheric effects and geometric distortions from the sensor. Only once these corrections have been performed, can the spectra be compared to existing laboratory measurements, the image spectra calibrated to the library spectra and the model applied to the whole image.

There are a number of limitations to using library spectra; in particular, it needs to be extensive. Not only must every species in the image be present but ideally, each should be present under different conditions. For example, there should be spectra for dark soils through to light soils, spectra for the same green vegetation growing on different backgrounds, with different stress conditions (little water, little nutrients) and so on (Bateson and Curtiss, 1996). Laboratory measurements for vegetation cover are especially difficult because the spectral response depends on intrinsic parameters which could not be registered in a spectral library such as canopy architecture and leaves' reflectance, the influence of which is not always predictable, and moreover vegetation response changes with time (Kerdiles and Grondona, 1995). Furthermore, sophisticated atmospheric modelling procedures for converting reflectance to radiance values and removing atmospheric effects are required (Craig, 1994). Models differ and resulting image spectra are not always similar to library spectra (Bateson and Curtiss, 1996). In addition, sensor degradation adds to the difficulties of modelling atmospheric effects (Kerdiles and Grondona, 1995). These are probably the reasons for which this method works best for soil measurements (Huguenin, 1997) and the problems have led some authors to state that laboratory reflectances are of limited value (Settle and Drake, 1993) and to establish other methods to determine mixture components, namely, identifying end-members directly from the image.

Image end-members can be identified in a variety of ways, the most common being simply from homogeneous areas which ground data identifies as *pure* or by plotting the spectra or principal components; the extremes of the plotted data are the end-members. For example, Bryant (1996) use the method developed by Smith *et al.* (1985) using principal components, to map salts on a playa surface from a Landsat TM image. When ground data is not available, end-members can be identified using a higher resolution image and then applied to the lower resolution image (Gong *et al.*, 1994; Cross *et al.* 1991;

Asner *et al.*, 1997). This is often the method used for AVHRR imagery for which, due to the size of the pixels, *pure* pixels are particularly difficult to find. Holben and Shimabukuro (1993) estimate the spectral response of vegetation, soil and shade for AVHRR pixels by regressing them against TM image pixels. In a slightly different approach, Oleson *et al.*, 1995 estimate cover type reflectances given the fractional areas of multiple cover types, calculated from high resolution Landsat TM data, in a coarse spatial resolution pixel. Multiple linear regression is applied to determine the mean reflectance of cover types.

The problem with image end-members is that they may not be *pure* as they may be linked to categories such as *Vines* or *Orchards* which themselves may be combinations of more fundamental reflectance spectra. It is also assumed that all materials within the image have sufficient spectral contrast to allow their separation and that the number and identity of each component can be defined in some way but this may not be and *pure* image end-members may not be identifiable (Sohn and McCoy, 1997; Gong *et al.*, 1994). This is particularly true as resolution decreases, for example to AVHRR resolution. The number and characteristics of end-members in a spectral data set are determined not only by the spectrally unique materials on the surface but also by illumination geometry and process (Bateson and Curtiss, 1996). Calculated end-member spectra, particularly if calculated automatically with no user input, may not be realistic; for example negative proportions may be derived, or data may not be fit uniquely. The situation may also arise where trees, for example, appear in more than half of the pixels on the ground but never actually constitute more than 35% of any single pixel in the image, thus making their identification as an image end-member impossible (Tompkins *et al.* 1997). If more spectral end-members than are present in the image data are identified, then the uncertainty in abundance estimates increases; if not all end-members are taken into account, then uncertainty in abundance estimates also increases (Smith *et al.*,1994).

Overall, linear unmixing models have two serious shortcomings. The first is that the assumption that the spectral response of a pixel is linearly dependent only on class proportions within the pixel can be erroneous (Jupp and Walker, 1997). Some authors have shown that the change in reflectance as a result of changes in proportions may be approximately linear in particular cases; for example Donoghue *et al.* (1995) identified three main mixing plains, water, vegetation and intertidal flats, when mapping an estuary. But although Jasinski and Eagleson (1990) demonstrated that for many semi-vegetated scenes, bare soil pixels orient themselves along a preferred soil line at the base of triangular red-infrared scattergrams and all the pixels falling on a line parallel to the soil

line may possess equal amounts of vegetation cover, the distance of that line from the soil line was not linearly proportional to the amount of vegetation but depended on the amount of vegetation and shadow and the magnitude of the reflectances. Mc Cloy and Hall (1991) found that the canopy reflectance of woody vegetation is closer to the plain of soils than is the response of green herbaceous material.

Non-linear mixing occurs when multiple scattering effects are considered. For example, a one layer model of vegetation above ground was shown to exhibit dramatically increased reflectance in the near-infrared due to multiple reflections between leaves and soil (Borel and Gerstl, 1994). Small, sub-pixel features may have an effect on the spectral response of a pixel that is not proportional to their area extent. Examples include fires and snow.

The second serious limitation of linear unmixing is that it is limited to ***n-1*** classes, where ***n*** is the number of channels of the sensor. This is sometimes referred to as the condition of identifiability (Kent and Mardia, 1988) as it is the requirement for a unique solution to exist. If it is assumed that one end-member is shade, then for the Landsat TM sensor, only four other end-members remain. A small number of end-members has the advantage that analysis is simplified but some materials will not fit a mixture of the few selected end-members and some image spectra will appear to be mixtures when in fact they are associated with different materials; errors will occur either as mis-identified materials or as incorrect fractions of end-members (Smith *et al.*, 1994).

A number of authors have suggested improved methods for end-member selection (Bateson and Curtiss, 1996; Bosdogianni *et al.*, 1997; Craig, 1994; Mathieu-Marni *et al.*, 1996; Tompkins *et al.* 1997). Improvements have also been suggested for the linear model, for example by suggesting that spatial relationships be taken into account (Mayaux and Lambin, 1995), by carrying out spectral unmixing locally rather than globally to overcome the ***n***-1 restriction (klein Gebbinck and Schouten, in print; Huguenin, 1997) or by adding simple post-processing non-linearities (van Kootwijk *et al.*, 1995). But although the linear unmixing model has been successful in a number of projects, results may show a lot of scatter (Thomas *et al.*, 1996) and inaccuracies which are thought to be a result of the model's limitations.

2.2.3 Summary

Although klein Gebbinck and Schouten (in print) suggest that linearity is the sensible physical basis which other methods lack, the assumption of linear mixing is not often correct and simply solving the linear equation is not a guarantee of a correct

solution. A poor fit may indicate an inappropriate selection of reference spectra, a non-linear response of the imaging instrument or non-linear mixtures of surface materials. Furthermore, using Landsat TM imagery, not more than five classes can be identified. Although with the event of hyper-spectral sensors this restriction may be overcome, alternative approaches have been suggested which overcome the limitations of the linear unmixing model.

2.3 Modified Maximum Likelihood and Other Fuzzy Algorithms

'Fuzzy' sets and logic (Klir and Yuan, 1995) are a branch of mathematics introduced by Zadeh in the 1960's (Zadeh, 1965). The theory of fuzzy sets is fairly involved from the mathematical point of view (Di Zenzo *et al.*, 1987) but its popularity has dramatically increased in the last five to ten years, even though it has at times been subject to controversy (Zadeh, 1978; Cheeseman, 1985; Puri and Ralescu, 1982). Several authors have commented on the confusion arising from the indiscriminate use of the term 'fuzzy' (Fisher, 1996; Lagacherie *et al.*, 1996). Unfortunately there has been a tendency in the remote sensing literature to use the term 'fuzzy' simply in contrast to 'hard' or 'pure' even though fuzzy mathematics are not used. In this thesis, preference is given to the term '*soft*' to dispel any confusion, although the terms used by authors will be respected when describing their work.

The algorithms grouped under this section all use some form of measurement of the distance between the pixel under consideration and class means, also called centroids. The fundamental assumption is that data that are similar are spectrally close to each other (Cannon *et al.*, 1986b); the more of a cover class a pixel contains, the more spectral characteristics of that class it has (Wang, 1990a). The main algorithms either use modified versions of the maximum likelihood classifier or a supervised version of the fuzzy c-means algorithm.

2.3.1 Modified maximum likelihood algorithms

The maximum likelihood classifier calculates means and variations of classes based on training data. It then applies these statistics to image or testing data pixels to calculate, for each pixel, its likelihood of belonging to any of the classes. In conventional classification, the pixel is assigned to the class with the highest likelihood value, also called *a-posteriori* probability. The likelihood of pixel $\boldsymbol{x}$ belonging to class $\boldsymbol{i}$, $\boldsymbol{L(i|x)}$, is given by Bayes' Rule:

$$L(i|x) = \frac{P_i \cdot P(x|i)}{\sum P_j \cdot P(x|j)} \qquad (2.3)$$

where P_i is the *a-priori* probability that x belongs to i and $P(x|i)$ is the probability density function for x belonging to i. If it is assumed that the data is normally distributed, $P(x|i)$ can be estimated using:

$$P(x|i) = \frac{1}{(2\pi)^{N/2}|V|^{1/2}} \cdot e^{-\frac{1}{2}D^2} \qquad (2.4)$$

where N is the number of sensor channels, V is the variance-covariance for class i, and D^2 is the Mahalanobis distance calculated by

$$D^2 = (x-\mu_i)^T V^{-1}(x-\mu_i) \qquad (2.5)$$

where x is the pixel vector, μ_i is the mean vector for category i and T signifies the transpose of the vector (Swain and Davis, 1978; Thomas *et al.*, 1987).

Likelihood values have been used to indicate the degree of certainty with which pixels are classified. For example, Van Deusen (1995) uses an algorithm which iteratively assigns pixels to classes based on their likelihood values. Pixels that are clearly associated with a given class, as indicated by a high likelihood value in one class and low values elsewhere, are used to influence the decision on allocating a class to their uncertain neighbours, as indicated by low likelihood values for every class. Novel approaches to analysing classification results are reported in Fisher (1994a and 1994b) where likelihood values are animated either visually or audibly to indicate the reliability of a classification. When used as a measure of uncertainty (Zhang and Hoshi, 1994) or similarity (Zhu, 1997), likelihood values were found to improve the classification accuracy. Although it can be hypothesised that misclassified pixels are likely to be mixed pixels and that therefore likelihood values can be used to indicate mixed pixels, without reference data no conclusion regarding sub-pixel information can be drawn from these particular experiments.

Reference data is available in the experiment reported by Foody *et al.* (1992) for example, continuing experiments reported in Wood and Foody (1989). The *a-posteriori* probabilities of pixels, as provided by the maximum likelihood classifier, are used to map a transect between wet and dry heathland from Airborne Thematic Mapper (ATM) data. The authors calculate relatively high correlation values between ground cover, as measured by quadrants, and *a-posteriori* values for fifteen test data points.

Wang (1990b) introduced the concept of weighting the mean and variance of the

maximum likelihood by fuzzy membership values. How much a pixel belongs to a class determines how much it contributes to the mean and covariance of that class. Thus for example, the mean of a category is calculated by the following equation, where μ_i^* is

$$\mu_i^* = \frac{\sum_{t=1}^{n} f_i(x_t) x_t}{\sum_{t=1}^{n} f_i(x_t)} \qquad (2.6)$$

the fuzzy mean, $f_i(x_t)$ is the fuzzy membership value of pixel x_t for class i and n is the total number of pixels. The fuzzy mean and fuzzy variance-covariance matrices are then used within the maximum likelihood algorithm in equation 2.4 and equation 2.5. The fuzzy membership values could be obtained from any methodology but Wang (1990a and 1990b) chooses to use the maximum likelihood values for each pixel. In effect, fuzzy membership values and a-posteriori probabilities are one and the same here. The fuzzy membership values of seven Landsat MSS pixels are found to be in agreement with visually estimated proportions from an aerial photograph. The fuzzy classification results are also 'hardened' for comparison with results from a traditional classification. 'Hardening' consists in assigning a pixel to the class with the highest fuzzy membership value, or *a-posteriori* probability. The overall fuzzy classification accuracy is found to be higher than that of the conventional classifier.

Maselli *et al.* (1995), basing themselves on the results by Wang (1990a and 1990b), estimate fuzzy membership values by a maximum likelihood procedure modified by the inclusion of non-parametric prior probabilities, calculated from simple frequency estimates. The aim of the experiment is to relate fuzzy membership values with the percentage of deciduous forest cover and with basal area values within afforested plots in the Apennines. Regression analysis is performed to test the relationship. Although the authors report good correlations, the ground data show that twelve out of twenty-two plots are 100% *pure* deciduous, seven of the plots contain less than 20% deciduous cover and only three have percentage covers in-between. The ground data therefore does not provide a good range of values and the results have to be seen in that context.

In a later experiment, Maselli *et al.* (1996) evaluate the capability of the modified maximum likelihood algorithm to estimate cover proportions in a general landcover classification. The data used has been degraded from a Landsat TM satellite image. Thus, the composition of pixels in the coarse resolution image is provided by the distribution of landcover classes of the high resolution data within the pixels. Training of the classifier

is carried out with Landsat TM pixels whereas testing is performed on the coarsened data. The relationship between the fuzzy membership values and class proportions is evaluated using correlation values and graphs. Good correlation values are found but the graphs show considerable scatter.

2.3.2 Fuzzy c-means

The fuzzy c-means algorithm was originally developed for the unsupervised classification of *pure* data. It is a non-parametric method, based on measuring the distance between pixels and classes, and weighting it according to fuzzy membership values, similarly to the modified maximum likelihood algorithms discussed above. The algorithm works by seeking the least square minimum to

$$J(U, V)) = \sum_{i}^{row} \sum_{j}^{col} \sum_{k}^{c} (f_{ijk})^m \cdot D^2 \qquad (2.7)$$

where f_{ijk} is the matrix of fuzzy membership values and D is some form of distance measure such as Euclidean or Mahalanobis described in equation 2.5; c is the total number of clusters and m is a weighting exponent. The algorithm essentially works by initialising c, m and f_{ijk} and class centroids, randomly assigning pixels to each class and then moving the pixels around so as to minimise the least squares error (Bezdek, 1984; Cannon *et al.*, 1986a).

Cannon *et al.* (1986b) present improvements to the algorithm, which require extensive 'tweaking' of the parameters, and apply it to the classification of cereal fields. However, the authors are not concerned with sub-pixel components and simply ignore the added information provided by the fuzzy outputs, by only extracting the class with the highest fuzzy membership value.

Fisher and Pathirana (1990) decide to investigate the relationship between ground cover proportions and the fuzzy membership values. Using supervised and unsupervised methods, the authors establish that seven landcover classes existed within their Landsat MSS urban scene. Proportions within pixels are estimated from aerial photographs and correlation values calculated. From the membership values, it would seem that most pixels are mixed although since the ground data is not described, it is difficult to tell the distribution of proportions across classes. The authors find that correlation values between estimate proportions and actual proportions are highest for well defined classes and lowest for poorly defined classes.

Foody (1992) uses the fuzzy c-means algorithm in supervised mode by providing

class centroids into the program. The experiment uses the same data as that reported in Foody *et al.* (1992). Different values of ***m***, the weighting exponent, are studied. Values close to one provide an almost conventional classification; values greater than one provide more fuzzy classification. Graphs of fuzzy membership values against actual percentage cover of dry heath reveal that the more fuzzy classification performs better.

A further example of the use of the fuzzy c-means algorithm is provided by Foody (1994) who uses a supervised version to perform an ordinal classification of tropical forest cover into 'large', 'intermediate', 'small', 'very small'. 'Large' describes pixels with more than 80% forest cover; 'intermediate' describes pixels with between 20% and 80% forest cover; 'small' describes pixels with between 2% and 20% forest cover; and finally 'very small' is applied to pixels with less than 2% forest cover. Landsat MSS data provides the composition for data degraded to approximate AVHRR resolution. The classifier is trained with twelve pixels of forest and twelve of non-forest to extract fuzzy membership values. Thirty-three pixels are then used to derive a regression relationship between the fuzzy membership values and sub-pixel cover. The performance is measured using thirty-two mixed pixels. Good correlation values are obtained. Estimated proportions are then used to produce a confusion matrix using the ordinal groups described above. Although not mentioned in the article, it is interesting to note that percentage cover is not evenly divided between the groups. This will certainly have an effect on the classification accuracy calculations, although what that effect may be is not clear without more information concerning the ground data.

2.3.3 Comparison of algorithms

One of the earliest attempts at correlating likelihood values with ground data proportions is reported in Marsh *et al.* (1980) who use what they call an approximate maximum likelihood technique. The method is only applicable to two-component mixtures; class proportions are proportional to the Mahalanobis distances between the reflectance of the pixel and each class mean, divided by the Mahalanobis distance between the class means. The authors compare the approximate maximum likelihood technique to a general weighted average equation and a linear regression technique. For the general weighted average, the proportion of a component is simply a function of the Euclidean distance between pixel and mean vector of the class divided by Euclidean distance between the two classes. Landsat MSS data is used to test the methods; aerial photographs provide the ground data from which proportions are calculated. Forty *pure* pixels for each class are identified and seventeen mixed. Eight mixed pixels are used for

testing the accuracy. The other nine are used for the regression analysis. Accuracies are compared using root mean square error (RMSE). The AML algorithms performs best although there is a tendency for dark vegetation to be overestimated and light soils to be underestimated. Results are confirmed using other data sets.

Foody and Cox (1994) compare a linear unmixing model and a regression model using fuzzy c-means estimated membership values. They conclude that both models can un-mix pixel composition although the linear unmixing model provides slightly higher correlation values.

In Foody (1996a), a discriminant analysis and a fuzzy c-means algorithm are applied to the same data, namely an ATM scene which has been coarsened so that the original data provides the composition of the degraded pixels. Three landcover types are identified, trees, asphalt and grass, and the classifiers are trained on five *pure* examples of each. The accuracy is tested on thirty-five pixels. The relationship between *a-posteriori* probabilities and class proportions is found to be weak in the case of the discriminant analysis but strong in the case of the fuzzy c-means.

Bastin (1997) compares a fuzzy c-means classifier, a linear mixture model and probability values from a maximum likelihood classifier. An unsupervised classification of a Landsat TM image into four classes, is regarded as the reference data and the image is coarsened to various scales. The results show that some classes present difficulties for each classifier, signifying that this maybe a feature of the classes rather than the algorithms. The classifiers perform to a similar accuracy although they appear to have different areas of strengths and weaknesses.

2.3.4 Summary

In summary, the experiments reported in the literature have shown that there are indications of a strong relationship between *a-posteriori* probabilities or fuzzy membership values and landcover proportions within a pixel. However, the maximum likelihood methods assume that data is normally distributed and the supervised version of the fuzzy c-means algorithms requires the input of class means from pure pixels. When mixed pixels are present, class data is unlikely to be normally distributed and in some cases, pure pixels may be difficult to identify. An alternative which makes no assumptions about the data and which may not necessarily require the input of pure pixels, is a neural network classifier.

2.4 Neural Networks

The interest in artificial neural networks, and more particularly multi-layer perceptrons, was revived in the late 1980's after the publication of research by Rumelhart *et al.* (1986). The success of MLPs in remote sensing classification problems no longer needs to be established as they have often been shown to perform at least as well as conventional classifiers (Howald, 1989; Benediktsson, 1990; Paola and Schowengerdt, 1994). Other types of networks have also been shown to be successful classifiers such as the Learning Vector Quantization (Hernanadez *et al.*, 1992) and the Self Organising Maps (Kohonen, 1989) but the most commonly used type in classification in general is the MLP which will be concentrated on here.

The MLP is simply a data driven classifier which is implemented to perform several calculations is parallel. A thorough description is provided in chapter III and at this stage it is only necessary to know that the penultimate stage of classification is a vector of output values from which the class with the maximum is attributed to the pixel offered as input. The main advantages of neural networks is that they are non-parametric and can easily include attribute data at the training stage. No model for the data is assumed as the relationship between inputs and outputs is extracted automatically from the examples provided. The main disadvantage of neural networks is that training can be slow, although on the other hand, the testing or classification phase is usually very fast.

Reservations concerning the use of neural networks in the remote sensing field mainly stem from their complexity. In fact, neural networks have more in common with statistical packages than is often assumed. They can be regarded as non-linear regression analysis tools (Cortez *et al.*, 1997). The vector of output values has been proven mathematically to provide *a-posteriori* probabilities under a series of assumptions that include infinite data sets and an appropriate architecture (Shoemaker, 1991) and training data that adequately represents the underlying probability density functions (Richard and Lippman, 1991). Irrespective of the activation function (Ruck *et al.*, 1990), neural networks can extract information about *a-priori* distribution and conditional probability from training data so as to approach unknown but true Bayes' rule asymtotically (Osman and Fahmy 1994; Wan, 1990). Consequently, when the approximation is accurate network outputs should sum to one (Lowe and Webb, 1991).

A number of 'fuzzy-networks' exist (Kosko, 1992) such as the Fuzzy ArtMap described in Gopal *et al.* (1993) and Carpenter *et al.* (1992) or those used by Lee and Wang (1994) and Uebele *et al.* (1995) for classification problems with fuzzy inputs. However, the purpose of these classifiers is ultimately to perform *pure* classification. It is

the implementation of the network that uses fuzzy concepts, they are not however applied to the classes. These networks are not therefore appropriate here.

As for the previously discussed modified maximum likelihood classifiers, some authors have used *a-posteriori* probabilities, this time derived from a neural network classifier, to indicate degrees of confidence in assignments (Bartl and Pinz, 1992). Foschi and Smith (1997) use a neural network for the identification of sub-pixel amounts of wooded vegetation. However, only one output is used with a continuous value to represent the presence or not of wooded vegetation. The value is taken as a probability that the pixel does contain wooded vegetation, it does not represent the proportion that wooded vegetation may occupy. Without the reference data to compare network output values with ground cover, only hypothetical conclusions about the relationship between the two can be drawn from these experiments.

Few articles report experiments to determine the relationship between neural network output values and ground cover proportions as this technique is at the forefront of sub-pixel information research. Moody *et al.* (1996) investigate the relationship with simulated data. Two data sets are used: the first contains simulated *pure* and mixed pixels; the second contains real *pure* pixels but simulated mixed pixels. Mixed pixels are linear combinations of the *pure* pixels. For the real data, a Landsat TM image is degraded and coarsened pixels are assigned the most frequent class according to the high resolution imagery. The response of the network is plotted in two-dimensional graphs of feature space, that is one band against another, and different shades of grey indicate the network output strength. The network consistently produces a maximum output for the largest sub-pixel class but the error increases with mixing. The correspondence between the second largest output signal and the second dominant class is weakest for almost pure and extremely heterogeneous cases. This is to be expected, since the secondary class in almost pure pixels may be too small to produce a strong signal in the pixel and in very heterogeneous pixels, the secondary class will not be very different in size from the other classes and therefore contribute to the pixel signal by a similar amount. According to the authors, it may be that in order to use non maximum outputs to indicate sub-dominant classes, the outputs first need to be adjusted based on interclass distances in multi-spectral space.

Foody (1996b) investigates the relationship between network output values and percentage cover using two data sets. The first consists of degraded ATM imagery from which three main *pure* classes are identified; the second consists of an AVHRR image with the percentage of forest cover within pixels is provided by a Landsat MSS image.

The neural network is trained on pure classes and tested on mixed pixels. The author finds that the network response is fairly hard. In other words, proportions in between the extremes are not well approximated. The node value for a class is close to one if that class covers more than 50% of a pixel and close to 0 if it covers less than 50%. Hypothesising that the sigmoid activation function of the network, that is the function which transforms input to output for each node of the network, may be influencing results, the author rescales output values to a linear function and results improve.

Foody (1997) use Landsat TM imagery to provide the composition of AVHRR pixels. Three classes are identified, River, Pasture and Forest. The particularity of this experiment is that since pure pixels are almost non-existent, the network is trained and tested with mainly mixed pixels. The target for the network uses scaled values of the proportions at every class. Network estimates are evaluated using correlation values and the total areal cover by a class. Statistically significant correlation coefficients are obtained although the graphs of estimated proportions against actual proportions, for each class, show some scatter.

2.4.1 Comparison of algorithms

Foody (1996a) uses coarsened ATM imagery to compare the abilities of a linear discriminant analysis, a fuzzy c-means and a neural network algorithm. Five pure pixels of each of the three *pure* classes are used to train the classifiers which are then tested on thirty-five mixed pixels. The effect of the sigmoid activation function is removed from the neural network output values. Performance of the algorithms is measured using measures based on the distance between predicted and actual proportions, and the difference between probability distributions. The graphs and the measures of accuracy indicate that the neural network and fuzzy c-means algorithms are more appropriate than the linear discriminant analysis.

Schouten and klein Gebbinck (1997) compare neural network identification of sub-pixel information with least squares and modified least squares algorithms for classification problems using three different sensors. Three or four end-members are identified within each problem and simulated mixed pixels are generated as linear combinations of the classes. The mixed pixels are used in the training and testing data sets which contain approximately 15,000 pixels each. The target type used for training the network is not reported. Performance of the algorithms is judged on the basis of plots of the frequency of difference between fractions. That is, for each pixel, the difference between the predicted proportion and the actual proportion is computed, apparently for

every class, and frequency graphs are plotted. The network performs best for each classification problem.

Warner and Shank (1997) use two sets of simulated data. The first, a two class problem, is fully synthetic, both the *pure* and the mixed pixels are simulated. The second, a four class problem, is partially synthetic: *pure* pixels are from a SPOT HRV image but mixed pixels are generated using a linear model. The network for the first data set is trained using only the *pure* pixels. The use of a modified activation function is evaluated and the performance of the network on the fully synthetic data is compared to a linear unmixing model. The linear unmixing model produces much lower errors on the fully synthetic data and the modified activation function performs better than the conventional sigmoid function. Two networks are trained on the second data set: the first uses only *pure* pixels, the second used *pure* and mixed pixels in the training file. For the network using also mixed pixels, the target for the network consists of scaled values of the proportions of the classes, between zero and one. The network trained with mixed pixels is able to better represent fuzzy boundaries in feature space.

Atkinson *et al.* (1997) use classified images of SPOT HRV sensor as reference data for AVHRR imagery. The identification of sub-pixel information from a linear unmixing model, a fuzzy c-means algorithm and a neural network are compared. The network is trained with a target of scaled values of class proportions. Training pixels number 215 for the network and testing pixels for all the algorithms number 108. Accuracy is evaluated using graphs of known percentage cover against predicted percentage cover. The linear unmixing model, trained on five pixels for each of the four or five end-members, performs particularly badly for this problem. The fuzzy c-means algorithm performed marginally better. The neural network was the most accurate estimator. These results may imply that the unmixing model and the fuzzy c-means algorithm are particularly sensitive to the end-member data that they are presented with. Considering that, particularly for AVHRR imagery, end-members are difficult to identify, this suggests that neural networks are advantageous in this case.

2.4.2 Summary

Investigating the relationship between neural network output values and class proportions has only very recently been carried out. Results show that there is a strong potential for using networks to identify sub-pixel information. Neural networks are particularly attractive as they are non-parametric and do not assume any relationship between individual components signatures and resulting mixture signatures. Furthermore,

it may be unnecessary for *pure* pixels to be used in the training stage. Neural networks compare favourably with other algorithms for the identification of sub-pixel information.

2.5 Data, Training Strategy and Accuracy

The results from the experiments reported in the literature described above, often show that there is reason to believe that there is a strong relationship between *a-posteriori* probabilities or fuzzy membership values and ground cover proportions within pixels. However, a number of issues have not been addressed.

2.5.1 *A-posteriori* probabilities and ground cover proportions

It has become common in the literature to use the concepts of uncertainty and mixing interchangeably. Fisher and Pathirana (1990), for example, explain that fuzzy membership values close to one show that it is very likely that the pixel belongs to the class and consequently, values in between one and zero show pixel proportions. In fact, there appears to be no theoretical proof that *a-posteriori* probabilities or fuzzy membership values which indicate the certainty with which a pixel can be assigned to a class should reflect class proportions. The *a-posteriori* probability that a pixel belongs to a class is the probability, based on *a-priori* probabilities and after using an algorithm, that the whole pixel belongs to that class. A probability of 0.3 of a class belonging to class 4, means that the pixel has 30% probability of belonging to class 4. There is no mathematical reason why a probability of 0.3 that the pixel belongs to class 4 should imply that 30% of the pixel is covered by class 4, even though it may seem intuitive. Campbell and Hashim (1992) insist that there is no theoretical justification for using probability or distance measures as proportions. The fact remains that empirically, as seen above, *a-posteriori* probabilities and fuzzy membership values have been shown to be related to ground cover proportions.

2.5.2 Data

Most of the experiments described in the previous section use classified high resolution data as reference data from which to calculate proportions for coarse resolution images: either the high resolution image is degraded to produce a coarse resolution image, or imagery from two different sensors are used, for example AVHRR and Landsat TM (Cross *et al.*, 1991; Maselli *et al.*, 1996; Foody, 1996b). Pixels in the high resolution imagery are considered to be *pure*. Although in some cases collecting reference data is not possible, this method ignores the fact that the classification of the high resolution image

will usually not be 100% accurate; indeed the accuracy with which the original image is classified is rarely reported. Since the reference image is not 100% correct, the proportions that are calculated from it will not be 100% correct either. Furthermore, in the case where an image is degraded, the problem is not necessarily realistic since pixel signatures from a low resolution image, as obtained independently from another sensor, are not simply some form of average of pixel signatures from a high resolution image. Even though some methods use more sophisticated models (Oleson *et al.*, 1995), they are still only a model. In the case where an independent high resolution image is used, georeferencing the two images may include some errors and consequently, the proportions that are calculated will not be accurate.

Some experiments use simulated data: sometimes both *pure* and mixed pixels are simulated but more often, *pure* pixels are taken from an image and mixed pixels are generated (Settle and Drake, 1993; Warner and Shank, 1997; Schouten and klein Gebbinck, 1997). Most commonly, mixed pixels are generated as linear combinations of the signatures of the *pure* pixels. This method presents a simplified problem to the classifiers which does not realistically mirror the problems that may be found in reality. Although useful for determining general trends, the methods make a series of assumptions which make it difficult to transpose the results onto a more realistic problem.

In addition, many of the experiments use small to very small numbers of pixels in the training and testing data sets (Bosdogianni *et al.*, 1997; Foody, 1992; Foody, 1996a; Wang, 1990a). This makes it difficult to draw any definitive conclusions that are statistically valid. The use of small data sets for training also render the use of statistical techniques such as the maximum likelihood algorithm somewhat questionable. The maximum likelihood is not a particularly robust technique and is really only reliable when its assumptions are met (Swain and Davis, 1978). These include normally distributed data and statistically significant data sets. Canters (1997) suggests that the strength of the relationship between class proportions and maximum likelihood *a-posteriori* probabilities will depend on the characteristics of the training data; with poor quality reference data or a strong deviation from normality, correlations may no longer be strong.

2.5.3 Training strategy

The training strategy for algorithms has not been discussed in the literature in any detail. Methods such as the maximum likelihood classifier, the fuzzy c-means or the linear unmixing algorithm require *pure* pixels. But as Canters (1997) suggests, it remains questionable that reliable conclusions on fuzzy membership can be drawn from the output

of a supervised classification procedure which generates spectral signatures from training pixels that are pure. Neural networks have the possibility of being trained with mixed pixels, and Wang (1990a) suggests that the requirement for homogeneous training sets is therefore less important. However, the articles which suggest this alternative have not usually compared results with training with pure pixels (Atkinson *et al.*, 1997; Foody, 1996b). Moody *et al.* (1996) chose training pixels that are neither extremely pure nor too mixed but do not discuss the reasons for this. Warner and Shank (1997) compare the partitioning of feature space between networks trained with synthetic pure pixels and networks trained with synthetic mixed pixels. They decide that the representation using mixed pixels is more accurate, but the results are qualitative rather than quantitative. According to Foody (1996), Pham and Bayro-Corrochano (1994) suggest that the back-propagation algorithm produces a network which can interpolate, but there has been no actual investigation as to whether this is true.

Furthermore, for the neural network classification, the format of the target which the network must aim for has not been investigated. The few articles that report a non-pure target simply use scaled values of the percentage cover (Atkinson *et al.*, 1997; Foody, 1997; Warner and Shank, 1997) but comparisons are not carried out with experiments using different target types nor is the possible influence of the target type discussed.

2.5.4 Accuracy

There have been numerous studies of methods for measuring *pure* classification accuracies (Congalton, 1991; Janssen and van der Wel, 1994) but the analysis of *soft* classification results is particularly difficult because of the increased dimensionality of the classification results. One of the simplest methods for measuring performance of an algorithm is computer time (Shimabukuro and Smith, 1991) but that provides no information concerning classification results. Not all methods are applicable to every problem. For example, Gopal and Woodcock (1994) develop accuracy measures which use fuzzy logic. However, they are only applicable when the outcome of classification is assumed to be *pure* but the ground data is fuzzy. Nevertheless, some of the issues, such as mismatches between class allocations, can be relevant.

Different graphical methods have been developed. The performance of the network for different proportions has been judged using, for example, plots of the frequencies of the difference between predicted and actual fraction (Schouten and klein Gebbinck, 1997); average error against proportions for different Bhattacharyya (separation index) distances (Warner and Shank, 1997); bar charts of the frequency of proportions obtained

for each method compared to the actual frequency of proportions (Bastin, 1997). These methods, however, calculate total or average values and do not show the distribution of results per pixel.

The most common type of plot is that of the predicted fractions against the actual fractions for each class (Foody and Cox, 1994; Thomas *et al.*, 1986). This type of plot shows the distribution of pixels for each class. However, the relationship between classes within a pixel is lost. For example, although an algorithm might have correctly predicted the percentage cover for class A in the pixel, it may be wrong for the proportion of class B within the pixel. Since the results for each class are represented on different graphs, this cannot be inferred from the plots.

For neural networks, but applicable to other methods, authors have used plots of pixels, band x against band y, where the response of the pixel is graded in grey scale. High response is usually white, low response is usually black (Moody *et al.*, 1996; Warner and Shank, 1997). These plots can show that response is usually highest where *pure* pixels are expected and lowest where mixed pixels are thought to lie. Again, however, the plots are class by class and the relationship between them within pixels is not obvious.

Fraction images are a similar technique for evaluating results, most commonly used in linear unmixing methods (Adams *et al.*, 1986; Fisher and Pathirana, 1990), which keeps the spatial relationship of pixels intact. For fraction images, one image per class is created. The intensity of each pixel in an image is the *a-posteriori* probability, fuzzy membership value or neural network output value for that class. These images are particularly useful for providing a spatial overview of the classification results. However, the analysis can only be qualitative and the relationship between classes within a pixel is not kept.

For quantitative analysis, one common method to analyse *soft* classification output is 'hardening' the results (Kent and Mardia, 1988). 'Hardening' consists of assigning pixels to the class which registers the highest *a-posteriori* probability, fuzzy membership value or network output (Wang, 1990a; Maselli *et al.*, 1996; Foody, 1992) and using a common confusion matrix to study the misclassifications. Essentially, this reverts to *pure* interpretations of the data and so is only useful for determining how a *soft* classification, rather than a *pure* classification, has affected overall accuracies; no *soft* information is provided. Actual and estimated total area estimates per class can be compared (Thomas *et al.*, 1996; Foody *et al.*, 1997; klein Gebbinck and Schouten, in print) but the problem with this method is that it is quite possible to have high overall accuracies but low individual accuracies as shown by Quarmby *et al.* (1992).

Most experiments use some measure which provides an overall accuracy value or an overall accuracy value per class. Measures that have been developed are usually based on the distance between the actual proportion of classes and estimated fractions. They include simple correlation values or Root Mean Square (rms) errors (Marsh *et al.*, 1980; Atkinson *et al.*, 1997); Euclidean-like differences between fraction, or between the probability densities of fraction estimates (Foody, 1996a; Foody and Arora, 1996); so-called Relative Probability Entropy (Maselli *et al.*, 1996) or cross-entropy (Foody, 1995).

Matrices have been developed which use some of these measures for the analysis of the distribution of results. For example, correlation values between *pure* classes can be displayed in a matrix (Moody *et al.*, 1996; Bastin, 1997). Fisher and Pathirana (1990) use matrices of the frequency of occurrence of fuzzy membership values which simply describe the data. They also use per class matrices of proportion and fuzzy membership value ranges. These matrices show the distribution of results within a class. A similar matrix is that used by Foody (1994), although ranges are assigned a name rather than a value.

Finally, the relationship between dominant and secondary classes, and the accuracies which would be obtained if misclassification between the two is considered correct, is analysed in a few articles. Alimohammadi (1994) found that the second class identified by a maximum likelihood classifier, that is the class which had the next-to-maximum likelihood value, makes the most significant contribution to classification errors. Moody *et al.* (1996) examine accuracies when the dominant class is assigned as a secondary class and vice versa. Canters (1997) produces tables of accuracy if misclassification to the second, to the second or the third, to the second, third or fourth position are considered accurate. Zhang and Foody (1998) suggest that results can be analysed by sorting fuzzy vector and ground proportions vector into decreasing order and comparing the classes at each position.

2.6 Summary of Chapter II

A review of the main results reported in the literature reveals that the most common algorithms used to identify sub-pixel information can be divided into three groups: linear unmixing models, modified maximum likelihood and other fuzzy algorithms, and neural networks.

Although linear unmixing models, discussed in section 2.2, can be successful, they have two serious limitations. The first is that they assume that the spectral signals from

pixel components mix linearly, whereas this may not always be the case, particularly for intimate mixtures. The second limitation is that they are restricted to ***n-1*** classes where ***n*** is the number channels of the sensor. For these reasons, alternative algorithms have been investigated.

So called *fuzzy* algorithms, discussed in section 2.3, mainly include modified maximum likelihood and fuzzy c-means classifiers. These algorithms use some form of measurement of the distance of a pixel from class centroids to estimate class proportions. Results show a strong potential for a relationship between *a-posteriori* probabilities or fuzzy membership values, and percentage cover. However, the maximum likelihood algorithm assumes that data is normally distributed, an assumption that is particularly unlikely for mixed pixels. The fuzzy c-means algorithm does not assume that data is normally distributed, but it requires, in supervised mode, input of class means. *Pure* pixels from which to estimate class means may not always be easily obtainable, especially for low resolution data.

The most recent studies, discussed in section 2.4, have investigated the use of neural networks for identifying sub-pixel information. Neural networks, specifically multi-layer perceptrons, are an attractive alternative to the other two groups of algorithms because they make no assumptions about the statistical distribution of data or about the way spectral signatures from pixel components may mix. Furthermore, they do not necessarily require *pure* pixels in the training stage.

Results reported in the literature suggest that neural network output values may be related to ground cover proportions. However, several issues arise from the methodologies used in the experiments described in the literature. In particular, data sets usually consist of very few classes and few pixels per class. Therefore, the data sets do not reflect the reality of the landscape and the complexity of landcover classification problems. Furthermore, the composition of mixed pixels is often calculated from classifications of higher resolution data or generated synthetically. The reference data against which classification results are compared may consequently be unreliable or simplistic. The training strategy for neural networks has not been investigated. Although some authors have used mixed pixels in the training data sets, the results have not been compared with using only pure pixels in the training data set. Furthermore, the target with which the network should be trained has not been examined. Using a target which reflects the proportions of classes in the ground data has not been compared to other target types. Finally, there seems to be no consensus concerning methods for measuring classification

accuracies and the techniques that are used have several shortcomings. In particular, most techniques examine results class by class which means that the relationship between classes within a pixel is lost. Quantitative methods that show where and how classes have been misclassified are also lacking.

The purpose of this thesis is to further knowledge concerning neural network interpretation of mixed pixels and their ability to identify sub-pixel information by addressing these issues.

Chapter III

THE MULTI-LAYER PERCEPTRON NETWORK

The network used in this thesis was implemented by the Environmental MAPping and modelling Unit (EMAP) of the European Commission's Joint Research Centre (JRC) of Ispra (Italy), at which the author was based. The algorithm was originally intended and used for *pure* classification of remotely sensed data (Kanellopoulos *et al.*, 1991; Kanellopoulos *et al.*, 1992; Kanellopoulos and Wilkinson, 1997). For the purposes of this thesis, it was modified and incorporated into a *soft* classification software package implemented by the author.

The network is a fully connected feed-forward multi-layer perceptron trained with a back-propagation algorithm. This is the type of network most frequently used in remote sensing classification problems (Dayhoff, 1990; Lisboa, 1992) and it has been shown to perform as well as, or better than, other algorithms such as the maximum likelihood algorithm (Howald, 1989; Bischof *et al.*, 1992), the minimum distance algorithm (Downey *et al.*, 1992) or multi-source statistical algorithms (Benediktsson *et al.*, 1990). Foody (1996a) and Atkinson *et al.* (1997) have shown the network to compare favourably with other techniques for *soft* classification problems.

This chapter provides an overview of the multi-layer perceptron network in the context of remotely sensed image data classification. First, the training, testing and classification processes are outlined. Then, details of the structure and training algorithm of the neural network are provided. The chapter continues with a review of some of the issues that must be considered when using this classifier. Finally, changes to the original computer code are outlined. Appendix A contains a full description of the software package and a listing of the source code.

3.1 Training, Testing and Classification

It is not necessary to know about the technicalities of the multi-layer perceptron network to understand its training, testing and classification phases. Within this section, the neural network will be regarded as a black box which takes input, processes it, and produces output.

3.1.1 Training

The network is trained with a supervised training algorithm. In supervised training, each pattern in the training data set consists of an input vector and a desired output vector, also referred to as the target. The target output is based on reference data and is assumed to be correct.

The learning algorithm is illustrated in figure 3-1. First, an input vector is presented to the network which processes it and produces an output vector. This output vector is then compared to the target output corresponding to the input vector. The difference between the two vectors, the desired or expected output and the estimated output, is calculated. The network configuration is changed so that the difference, or error, will decrease for this pattern. The next input vector is then presented to the network. After all the patterns have been presented once, one iteration has been performed and a mean error is calculated for the system.

Training is finished when a fixed number of iterations of the training data set have been performed or a minimum mean threshold error has been reached. At this point, the configuration of the neural network system is fixed. Testing and classification can now take place.

3.1.2 Testing

Testing is carried out to measure the performance of the network. In this phase, patterns from a testing data set are presented to the network. The network computes an output vector from each input vector. In *pure* classification problems, the class to which the network assigns the pixel is determined from the output vector. The output class is compared to the target class, provided by the reference data, to which the pixel belongs. The simplest measure of performance is a sum of the correctly classified pixels divided by the total number of pixels in the testing data set.

3.1.3 Classification

Once the network has been trained and tested, an image can be classified. For each pixel in the image, the network computes an output class to which the pixel is assigned. The final product is a classified image. The accuracy of classification is assumed to be that calculated in the testing phase.

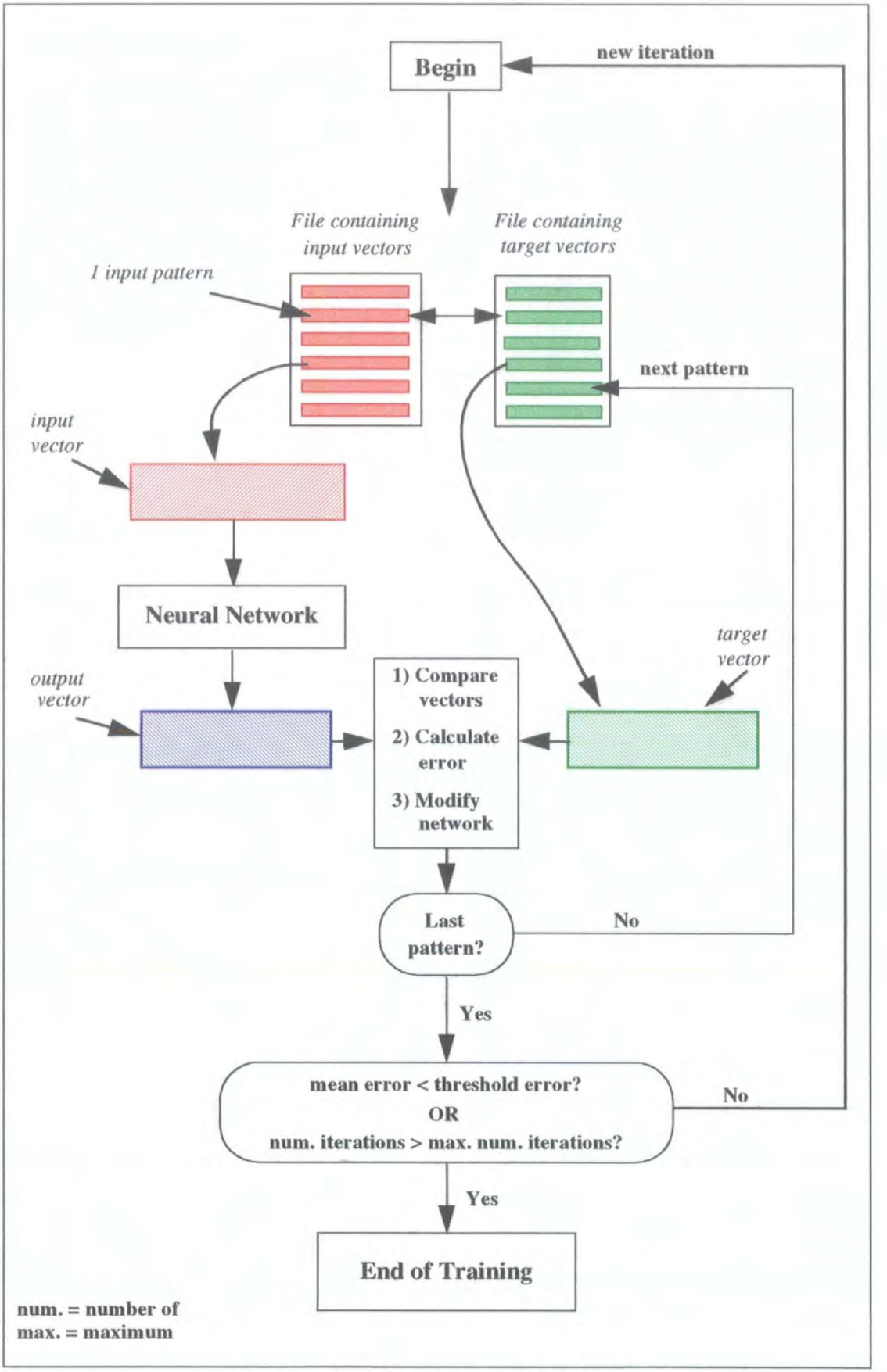


Figure 3-1. Training procedure for the feed-forward multi-layer perceptron

3.2 Network Structure and Training Algorithm

3.2.1 Network structure

At the simplest level, a neural network is a software program which implements a specific set of mathematical equations. The main characteristic of the program is that it performs parallel non-linear mapping from input to output (Works, 1992). The logical flow of the software program is represented by groups of processing units. Each processing unit, or node, takes an input value, **I**, applies a function to it, **f(I)**, and produces an output value, **O**. The sequence 'Input, Node, Output', shown in figure 3-2, is often called a neuron.

Figure 3-2. One neuron

Neurons can be combined in two ways. With the first method, illustrated in figure 3-3, the output values from a group of neurons are linearly combined and the overall result becomes the input to a new neuron. The connections between each neuron and the new neuron carry a value called a weight, or connection strength. The overall input to the new neuron is calculated by taking the sum of the product of each input value and its respective weight. The processing unit of the new neuron can then apply a function to the overall input pattern and produce a new output value.

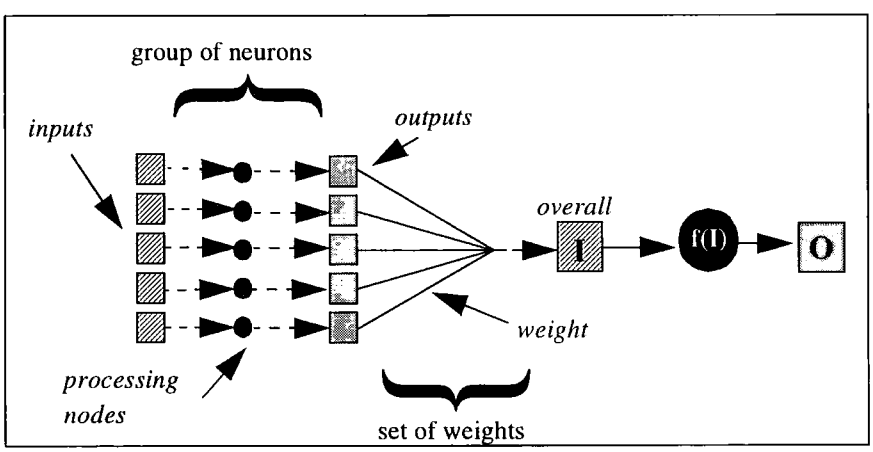


Figure 3-3. First method for combining neurons

The second method, illustrated in figure 3-4, uses the fact that different sets of weights, separately applied to the same group of neurons, are equivalent to the one group of neurons with different sets of weights applied in parallel. When several neurons are combined in both ways, a network is obtained. To simplify diagrams and explanations, it is common to represent a whole neuron by showing only its node as depicted in figure 3-5.

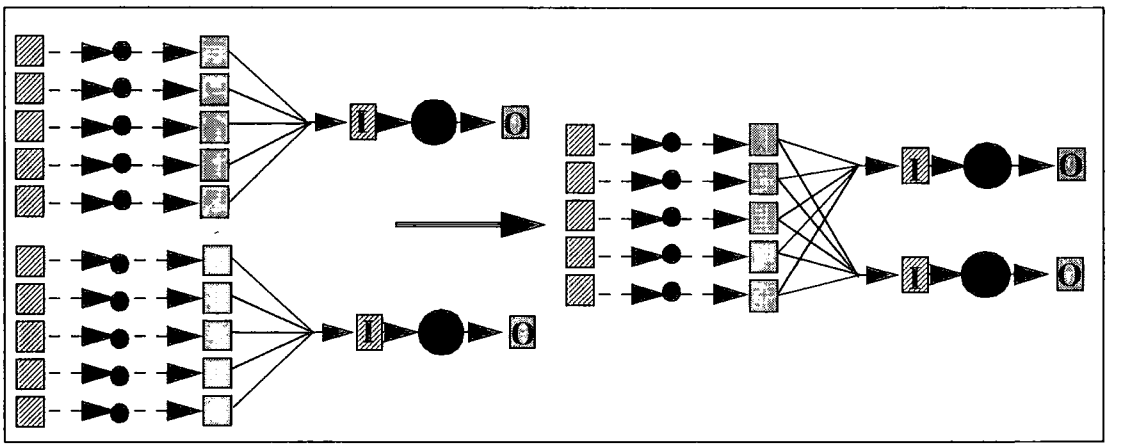


Figure 3-4. Second method for combining neurons

Nodes at the same level of connection belong to the same layer. It is common to represent an input vector from a pattern as a set of l neurons, where l is the number of components of the input vector. Each node receives as input its corresponding input vector value. The group of neurons is often termed the input layer and is the first layer of the network. No processing takes place in this layer, that is $f(x) = x$, and the outputs from the nodes are simply the raw input values. For this reason, some authors prefer not to represent the input values in this way (Kung, 1993). The last layer in the network, called the output layer, is generally the last layer of processing units. It is understood that this layer produces an output vector, whose elements are the individual output values, which is not represented. Nodes which lie in between the input layer and the output layer belong to hidden layers and are called hidden nodes. Thus, a network such as that depicted in figure 3-5, is termed a three layer network and consists of an input layer, a hidden layer and an output layer (which produces a vector of output values).

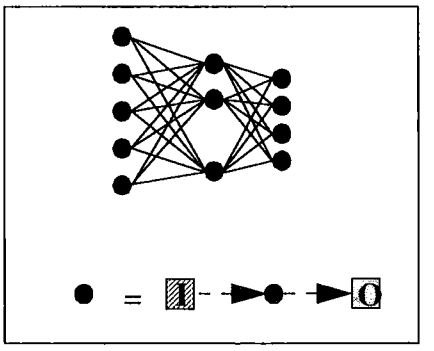


Figure 3-5. Illustration of a network

The simplest network is fully connected and feed-forward. In a fully connected network, all nodes in a layer are connected to each node in the layers immediately above and below (shown as right and left in figure 3-5) but not to nodes from the same layer. When information is only propagated in one direction, from the input layer to the output layer through the hidden layers, the network is called feed-forward. It is the simplest type

of multi-layer perceptron because at any instant, the output of the network only depends on the current input pattern and weights (Works, 1992; Bishop, 1995). Networks exist with more complicated structures such as missing connections, connections which skip layers or lateral connections (Hush and Horne, 1993), but the fully connected network is the most commonly used (Dayhoff, 1990)

3.2.2 Training algorithm

The training algorithm used for the multi-layer perceptron is a back-propagation algorithm whose basic logic was outlined in section 3.1.1. In this section, it will be explained in more detail, based on the description given in Rumelhart *et al.* (1986b). The discussion will be based on figure 3-6, a three layer, fully connected, feed-forward multi-layer perceptron containing l input nodes, m hidden nodes and n output nodes, but it is applicable to a multi-layer perceptron with more hidden layers. Nodes in the input layer I, are identified by the subscript i; nodes in the hidden layer J, are identified by the subscript j; nodes in the output layer K are identified by the subscript k. Weights between nodes are identified by the subscripts of the nodes they connect, for example w_{ij} for weights connecting nodes in layers I, the input layer, and J, the hidden layer.

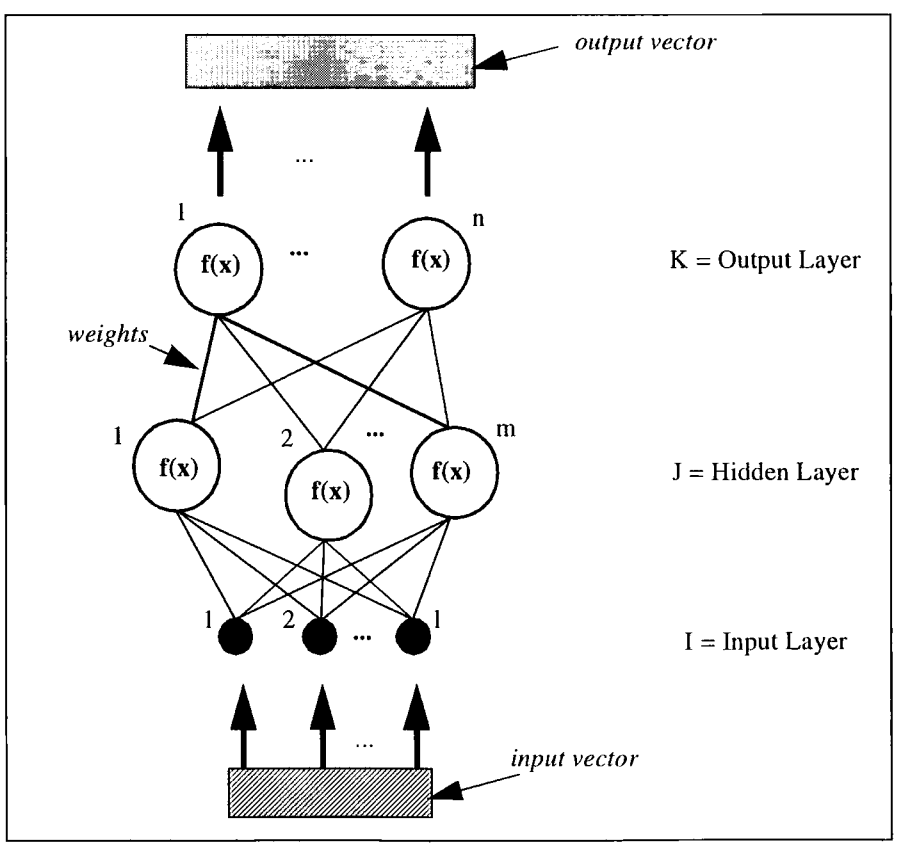


Figure 3-6. Generic feed-forward, fully-connected, three layer multi-layer perceptron

As mentioned in section 3.2.1, a node, also sometimes called short term memory, takes an input, applies a function to it, and produces an output. The function is usually called the activation function and the output of the neuron is referred to as the activation value (Simpson, 1990), output strength (Fisher 1994a), firing (Amari, 1993) or simply output value (Bishop, 1995). It is not necessary to know the details of the function at this stage.

Before applying the back-propagation algorithm, the weights of the network, sometimes also referred to as long term memory, are randomly initialised. Then, the information present in the input pattern is conveyed to each node in the hidden layer through a linear combination of the outputs from the input layer. In other words, the overall input, I_{ovr} , to each node in the J^{th} layer, therefore called I_{ovrj} , is the sum of the product of each output from the I^{th} layer, O_i , and the corresponding connecting weight, w_{ij} . Sometimes a bias term is included in the equations but this can be treated as a special case of weights from an extra input equal to one (Bishop, 1995).

$$I_{ovrj} = \sum_{i=1}^{l} w_{ij} O_i \qquad (3.1)$$

Once the overall input to each node in layer J has been calculated, it is transformed by the activation function $f(I_{ovrj})$ and the output of each node of the J^{th} layer, O_j, is computed.

$$O_j = f(I_{ovrj}) = f\left(\sum_{i=1}^{l} w_{ij} O_i \right) \qquad (3.2)$$

The output values from the J^{th} layer are then multiplied by the relevant weights, w_{jk}, and summed to produce the overall input to each of the nodes in the output layer K, using equation 3.1. Similarly, the output strengths for layer K, for each node, are calculated using equation 3.2. At this stage, an output vector of length n has been produced and the feed-forward stage of the back-propagation algorithm is complete for the input pattern.

Now the output vector is compared to the target vector which corresponds to the input vector. The difference, or error, can be calculated in several ways and authors disagree as to whether there is little effect on the accuracy (Richard and Lippmann, 1991) or whether results differ (Ripley, 1993). The sum of squares error function is the simplest (Bishop, 1995) and most typically used (Simpson, 1990). Let t represent the target vector and o the output vector from the network, then the error between the two, for pattern p, is

expressed by equation 3.3, where k is the index of the output node.

$$E_p = \frac{1}{2} \sum_{k=1}^{n} (t_k - o_k)^2 \qquad (3.3)$$

Consider figure 3-7.a which illustrates an arbitrary function which is differentiable. The first derivative of a function $f(x)$ is an expression of the rate of change of the function $f(x)$ with respect to x. It is the slope of the tangent to each point of the curve. When the function is at a minimum or maximum value, the tangent is parallel to the x axis, the slope is zero and therefore the first derivative is zero as shown in figure 3-7.b. To differentiate between a maximum and a minimum value, the second derivative of the function is taken. The second derivative is negative when the original function has a maximum at that point; whereas the second derivative is positive when the original function has a minimum at that point, as shown in figure 3-7.c.

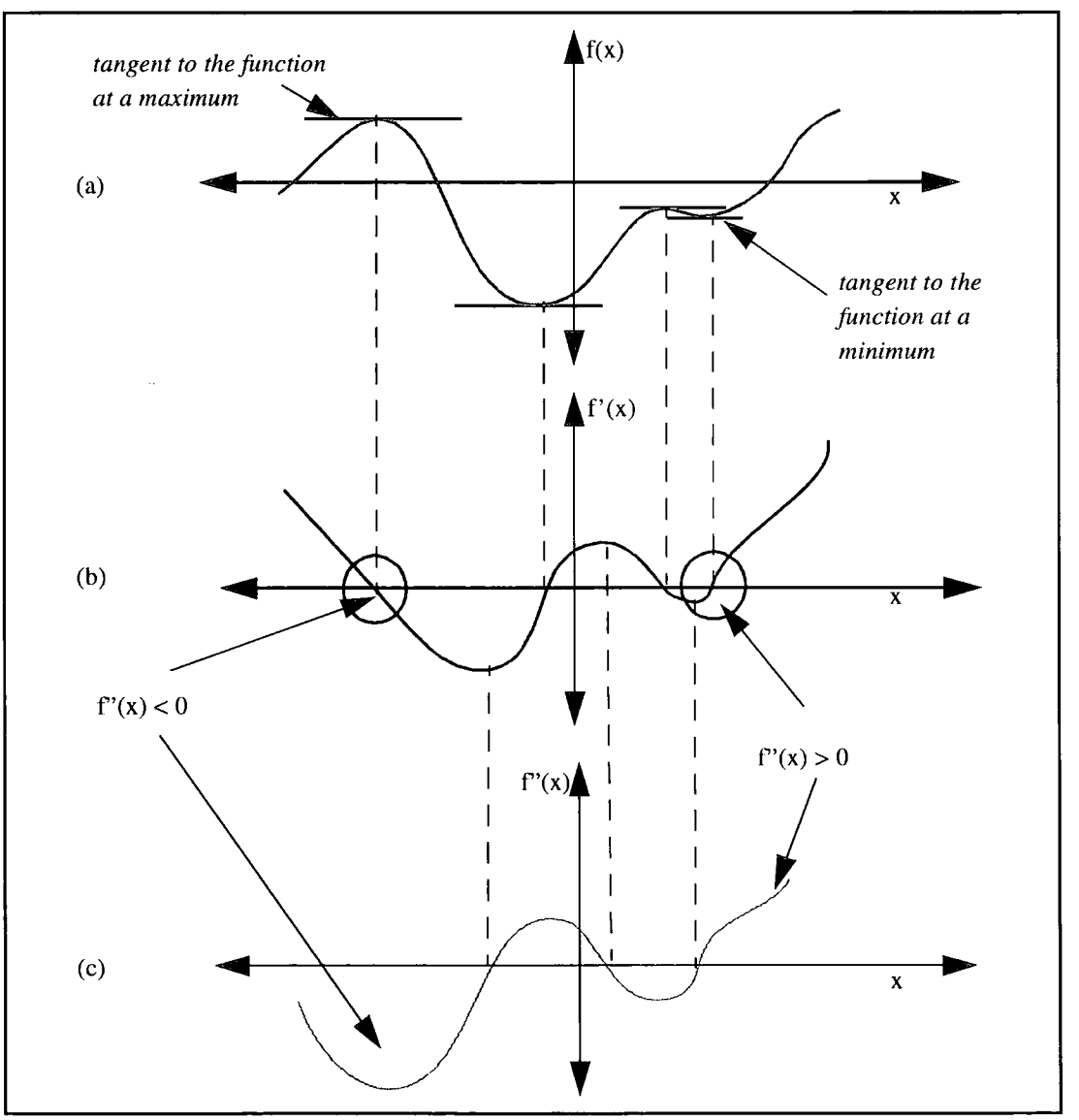


Figure 3-7. (a) Arbitrary function; (b) first derivative of the function; (c) second derivative of the function

Consequently, to find a configuration of the weights such that the error is at a minimum, changes to the weight values must be proportional to the derivative of the error with respect to the weights. The change in the weight value for a connection between node j in the hidden layer and node k in the output layer, between iterations t and $t+1$, is therefore computed by

$$w_{jk}(t+1) - w_{jk}(t) = \Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \qquad (3.4)$$

where w_{jk} represents the weight value between node j in the hidden layer and node k in the output layer, η is a constant of proportionality and is called the learning rate; Δ means change and $\frac{\partial}{\partial}$ means partial derivative.

Using the chain rule, the derivative can be written as the product of two elements. The first is the change in the error as a function of the change to the overall input to the node. The second is the change to the overall input to a node, as a function of the change to a particular weight connected to that node.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial I_{ovrk}} \times \frac{\partial I_{ovrk}}{\partial w_{jk}} \qquad (3.5)$$

From equation 3.1:

$$\frac{\partial I_{ovrk}}{\partial w_{jk}} = O_j \qquad (3.6)$$

Define

$$-\frac{\partial E}{\partial I_{ovrk}} = \delta_k \qquad (3.7)$$

Substituting equation 3.6 and equation 3.7 into equation 3.5 and subsequently into equation 3.4:

$$\Delta w_{jk} = \eta \delta_k O_j \qquad (3.8)$$

This is known as the generalised delta rule. From equation 3.7, and using the chain rule:

$$\delta_k = -\frac{\partial E}{\partial I_{ovrk}} = -\frac{\partial E}{\partial O_k} \times \frac{\partial O_k}{\partial I_{ovrk}} \qquad (3.9)$$

From equation 3.2:

$$\frac{\partial O_k}{\partial I_{ovrk}} = f'(I_{ovrk}) \qquad (3.10)$$

If node j under consideration is an output node, then from equation 3.3:

$$\frac{\partial E}{\partial O_k} = -(t_k - o_k) \qquad (3.11)$$

and therefore:

$$\delta_k = (t_k - o_k) f'(I_{ovrk}) \qquad (3.12)$$

If, on the other hand, the node under consideration is not an output node then, in equation 3.9, using the chain rule:

$$\frac{\partial E}{\partial O_k} = \sum_j \frac{\partial E}{\partial I_{ovrj}} \times \frac{\partial I_{ovrj}}{\partial O_k} \qquad (3.13)$$

From equation 3.1:

$$\sum_k \frac{\partial E}{\partial I_{ovrj}} \times \frac{\partial}{\partial O_k} \sum_i w_{jk} O_k = \sum_k \frac{\partial E}{\partial I_{ovrj}} \times w_{jk} \qquad (3.14)$$

From equation 3.7:

$$\sum_k \frac{\partial E}{\partial I_{ovrj}} \times w_{jk} = -\sum_k \delta_k w_{kj} \qquad (3.15)$$

and therefore, substituting into equation 3.9, for non-output units:

$$\delta_j = f'(I_{ovrj}) \times \sum \delta_k w_{kj} \qquad (3.16)$$

The error signal for hidden units is thus determined recursively in terms of the error signals from the output units to which they are connected and the weights between them.

The backward phase of the back-propagation algorithm therefore consists of the following steps:

1- calculate the error signal for each output node;

2- calculate the weight changes for the connections which feed into the output;

3- calculate the error signals for the hidden nodes.

Since the aim of the back-propagation algorithm is to minimise the overall error of the system, that is the error over all patterns p, a mean error E_p is given by

$$E_p = \frac{1}{2p} \sum_p \sum_{k=1}^{n} (t_k - o_k)^2 \qquad (3.17)$$

The delta rule is said to be a close approximation of a gradient, or steepest, descent procedure, even though weights are updated after each pattern; if the learning rate is small enough, the difference is negligible (Rumelhart *et al.*, 1986b).

In a general sense, learning is defined as any change in memory over time not equal to zero (Simpson, 1990). Since the weights and the processing units can be considered to contain the memory of the neural network system, modifying the weights is termed learning (Le Cunn, 1987). The system is inherently parallel because several units can be updated at the same time. The computations of changes to the weights and nodes are extremely complex and time consuming; nevertheless, an example of the calculations involved for a simple problem can be found in Aleksander and Morton (1992).

3.3 Using the Network

This section will describe some of the issues that must be considered when using the network. As is explained in this section, choices for the parameters that are set by the user are mainly a case of trial and error. Chapter V describes experiments that were carried out to test the sensitivity of the network for each parameter.

3.3.1 The activation function

From the derivation of the back-propagation algorithm, the activation function, sometimes also called threshold (Wilkinson, 1997) or squashing function (Simpson, 1990), must be chosen to be a differentiable, and therefore continuous, monotonic (non-decreasing), semi-linear function. Sigmoids are commonly used and a hyperbolic tangent function *tanh* was used in this implementation of the network:

$$f(I_{ovr}) = m \cdot \frac{e^{-kI_{ovr}} - 1}{e^{-kI_{ovr}} + 1} \qquad (3.18)$$

where $f(I_{ovr})$ is the activation function; I_{ovr} is the overall input to a node; k and m are constants. The function is illustrated in figure 3-8.

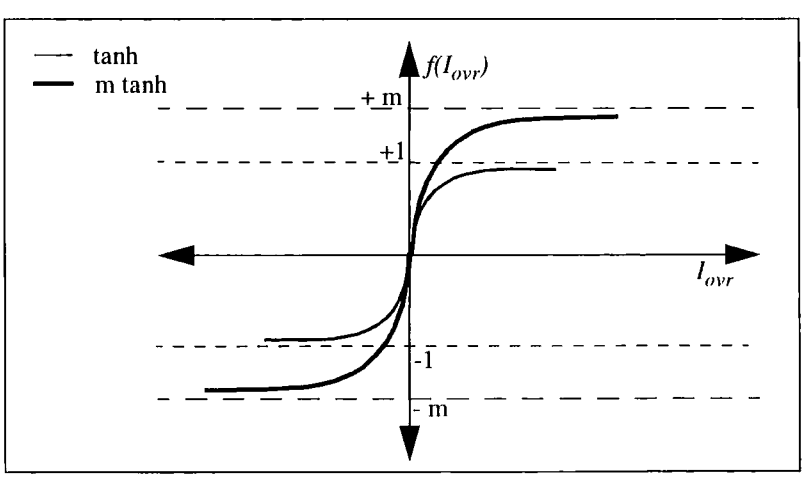


Figure 3-8. tanh and mtanh functions

This activation function has a range of [-1,1] and aims to produce output that is close to +1 or -1 by enhancing or suppressing the overall input to the node. For an unscaled *tanh* function (that is, without **m**), it would take an infinite amount of time to approach +1 and -1. If *tanh* is scaled by **m**, +1 and -1 can be reached in a less than infinite amount of time as illustrated in figure 3-8. It is also possible to choose the targets of the activation function in a smaller range than [-1,+1]. For example, for a sigmoid activation function with a range [0,1], the targets can be chosen to be 0.1 and 0.9 (Benediktsson *et al.*, 1990; Paola and Schowengerdt, 1995b). Values of **m** and **k** of 1.7 and 4/3 respectively have been shown to have practical benefits (Fogelman-Soulie, 1991; Bottou, 1991), the details of which lie beyond the scope of this thesis. These were the values that were used in the implementation of the network. They imply that neural network output values will lie between ~+/-1.7. The activation function is a feature of the implementation and cannot be modified by the user.

3.3.2 The weights

As shown in section 3.2.2, the change in weights is proportional to the derivative of the activation function. In the *tanh* function, the derivative is steep in the most linear part of the curve, in the centre. This implies that weight values in that region will change rapidly. On the other hand, weights will change little when values approach the desired values since the derivative in that part of the function is asymptotic.

The weight adjustment is proportional to the error value of the target node and the output value for the input node. Thus, if the error of a node is large, the change to its incoming weights is large. Similarly, if the output value of an incoming node is small, the weight adjustment is small and *vice-versa*. In other words, there will be a larger adjustment of the outgoing weights of a node with a high activation value (Dayhoff, 1990). Since weights control whether a signal should be amplified or reduced, they can be referred to as excitory or inhibitory, respectively (Simpson, 1990).

Since the error is proportional to the weight values, all hidden nodes connected to output nodes will get identical error signals if all the weights are initialised with equal values. Since weight change is dependent on the error signal, all the weights will change by the same amount and weights will always be the same. If weights are initialised to values which are too large, then they require large changes and training is unlikely to succeed. Thus, weights should ideally be small and initialized randomly. Results are improved if the weights are uniformly distributed (Yoo and Pimmel, 1994). Under these conditions, changes in weights produce a change in output approximately equal to the

range of weights (Kanellopoulos and Wilkinson, 1997) and training is more stable.

To understand the effect of weight changes, it is easier to consider a system with a linear activation function and one weight as shown in figure 3-9. In this case, the error equates to a simple quadratic. Its derivative is given by the straight line ***(t-o)***. Let the minimum error occur at weight value w_o. If the weight is currently at w_a, then it should be increased to reach w_o. If on the other hand it is at w_b, then it should be decreased.

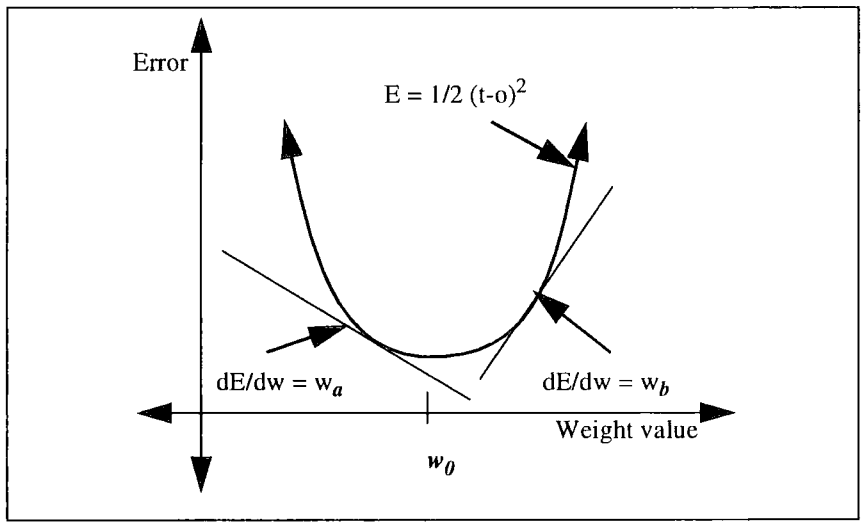


Figure 3-9. Possible weight situations for a system with a linear activation function and one weight

Weight space for a system containing hidden nodes and non-linear activation functions is more complex. Weights can be interpreted as a set of hyperplanes partitioning the input space (Fierens *et al.*, 1994) and can be viewed as a series of troughs and crests, the depth and height of which are the mean error at those points. From its similarity to the gradient descent procedure, the back-propagation algorithm can be expected to be caught in local minima (Rumelhart *et al.*, 1986a). A local minimum is a point in the weight space where the error is at a minimum, but not at the overall minimum of the function. Authors disagree as to whether this problem is rare in practice (Rumelhart *et al.*, 1986a) and can easily be solved by increasing the degrees of freedom, that is, the number of weights (Le Cunn, 1987), or occurs more frequently than imagined (Ripley, 1993).

In theory, it is possible to update weights either after each pattern or after each complete iteration, that is the presentation of all the patterns in the training set. The former is termed on-line training while the latter is termed off-line training or also, data adaptive or block adaptive respectively (Kung, 1993). Opinions differ as to which is the more efficient. Some authors advocate blocked back-propagation as improving speed of convergence and accuracy (Heerman and Khazenie, 1992; Liu and Xiao, 1991) and guaranteeing a decrease in mean square error from one iteration to the next (Paola and

Schowengerdt, 1994). Others counteract that on-line training is a local cost gradient which makes it more likely to enable the network to escape from local minima by updating weights more frequently (Fogelman-Soulie, 1991; Kanellopoulos *et al.*, 1993; Bottou, 1991) even though off-line methods may be more robust (Kung, 1993). The back-propagation algorithm used in the thesis implements on-line training.

3.3.3 The learning rate and the momentum term

From equation 3.4, the learning rate is a constant of proportionality. It is set by the user and influences the size of the step taken from one weight position to another. The most common cause of oscillation in a network is an improperly selected learning rate (Heerman and Khazenie, 1992). If the learning rate is small, the network takes a long time to minimize the error although it consistently follows the correct direction and is therefore eventually likely to converge if the weights do not get trapped in local minima. If the learning rate is large, large step sizes are taken which allow the network to step 'over' local minima and learning is fast but oscillations can be produced, resulting in an unstable network. Learning rates usually range between 0.1 and 1.0.

In addition to the learning rate, it is possible to add a term to the equation such that:

$$\Delta w_{ij}(t+1) = \eta \delta_j O_j + \alpha \Delta w_{ij}(t) \qquad (3.19)$$

where t is the iteration number and α is a constant. The second term of the equation is called the momentum term and is used to dampen the effects of the learning rate and thus reduce possible oscillations between two points by adding a fraction of the previously calculated weight change; this can speed up convergence.

The learning rate and momentum term can be kept fixed throughout training or can be reduced at fixed intervals. Paola and Schowengerdt (1994) for example, change the learning and momentum rates. If the error of the current iteration is less than the previous error, values of the rates are decreased, otherwise they are increased. As the authors argue, this procedure diminishes the importance of the original choice by the user. With on-line training, the momentum term is unnecessary (Fogelman-Soulie, 1991) and it was not implemented in the software. Unless otherwise specified, the learning rate was set at 0.1 for all the experiments described in this thesis.

3.3.4 The architecture

The number of input and output nodes of a network are determined from the available data and the classification problem; the number of hidden nodes is usually ascertained through trial and error although some authors have sought more formal rules

(Baum and Haussler, 1989). Pixels are represented by vectors of information which are used to separate the pixels into different clusters or classes. The vectors are the input patterns for the network. There are as many input nodes as there are vector components. Vector components can be channel values from one or several sensors (Kanellopoulos *et al.*, 1994b) and any value extracted from ancillary data for that pixel, such as an elevation value, a GIS value or a texture index, as illustrated in figure 3-10. The pixels are drawn from one data file whose format is as follows. Each row contains the input vector for one pixel and each column contains the information for one vector component, always in the same order.

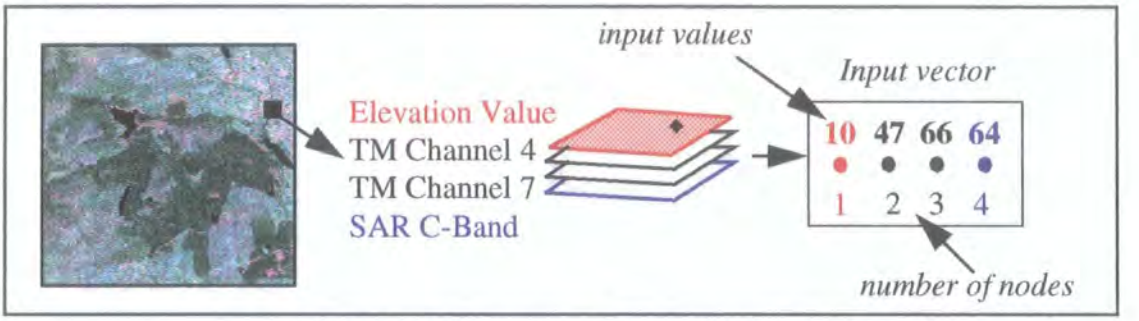


Figure 3-10. Illustration of the creation of an input vector

In the work presented in this thesis, no ancillary information was used. The input vector always contained all the channels from the Landsat TM sensor except for its thermal band which has a lower spatial resolution. Techniques such as principal component analysis are not always optimal with regard to class separability (Benediktsson and Sveinsson, 1997) and using all the channels of the sensor has been shown to provide the best accuracy of classification for some problems (Foody and Arora, 1997).

The number of nodes in the output layer is determined by the target vector. In *pure* classification problems, it is usual to use a vector of length ***n*** where ***n*** is the number of *pure* classes that have been defined for the problem, as illustrated in figure 3-11. The class to which the pixel belongs is usually signalled by a +1, shown in red, and the incorrect classes by -1 or 0. The choice of maximum and minimum are determined by the activation function.

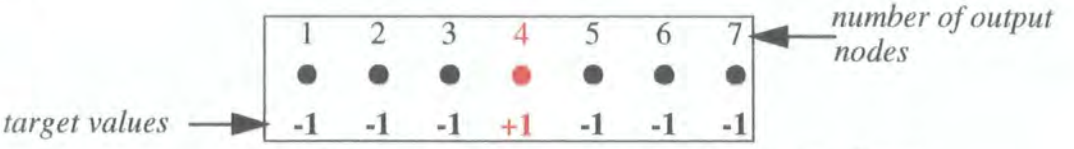


Figure 3-11. Pure target vector for class four of seven classes

For *soft* classification, the output vector is also a vector of length ***n*** where ***n*** is the number of *pure* classes that have been defined for the problem. However, the target values

are not necessarily +/-1 or 0 and 1. They may, for example, represent the ground cover by each class and lie in the range [0,1], the percentage cover having been scaled between these values (Atkinson *et al.*, 1997; Foody, 1996a). This will be discussed in more detail in chapter VII. In any case, for *pure* and *soft* classification, the output layer contains the same number of nodes as there are components of the target vector.

Whereas the input and output layers are defined by the characteristics of the classification problem, the number of hidden nodes are determined through trial and error. If there are too few hidden nodes, the data will not be adequately represented internally (Hepner *et al.*, 1990) and the network will not possess sufficient capacity to separate categories. On the other hand, if there are too many hidden nodes, the network will learn the training patterns but will not have the ability to generalize; there will be overfitting and poor interpolation (Kung, 1993).

Although it is impossible to train a network with one hidden layer to the same degree as a network with two hidden layers (Dreyer, 1993), one hidden layer is sufficient for most problems (Hepner *et al.*, 1990). The number of nodes in the hidden layer is a matter of trial and error and although authors have made suggestions, there are no rules. Increasing the complexity of a network increases the time taken to train the network. For this reason, all the experiments described in the thesis were carried out with networks with one hidden layer. Throughout the thesis, the convention will be to refer to a network as 6-13-7 for example, meaning 6 input nodes, 13 hidden nodes and 7 output nodes.

3.3.5 The datafiles

Running the neural network requires two data sets, one for training and one for testing. The data file is randomised and then the pixels are divided into a training and testing data set. Alternatively, pixels are divided into training and testing data sets and then the training file is randomised. Training patterns must be randomised before being presented to the network. The network adapts its weights over time. If the last patterns in an iteration are not randomised and thus are all from the same class, the network will adapt to represent this cluster, effectively 'forgetting' the previous patterns. In the testing phase, the network weights are fixed and it simply produces an output for each input pattern that it is presented with. It is therefore unnecessary to randomise the testing set.

The number of patterns in the training set is not fixed. For a general classification problem, Swain and Davis (1978) suggest 30 x number of classes x number of channels; Foody *et al.* (1995b) recommend at least 30 x the number of input features; Fogelman Soulie (1991) indicate that a network with W weights requires W/e training patterns to

yield an error less that e. Again, there are no rules. The number of patterns should be sufficiently large to adequately represent the variance of each cluster.

The number of patterns per class in the training and testing data sets depends on the classification problem. The number of patterns per class can provide an *a-priori* weighting of the importance of each category (Foody *et al.*, 1995a) and the neural network may be biased in favour of classes which are the most frequent in the training data set (Lowe and Webb, 1991). If no classification is carried out, the number of pixels per class in the training set can be equal for all classes. In this way, the testing accuracy provides an unbiased estimate of the generalisation capabilities of the network for each class. On the other hand, if classification is to take place, it is recommended that the number of patterns in the training set reflect the distribution of the classes in the image to be classified. The number of patterns in the testing file need not be the same as for the training file, but they should be in similar proportions for each class so that the accuracy calculations are not influenced by the number of patterns. The number of patterns per class should provide an adequate overview of the variability of the class.

The raw input data should be processed to make it centred and of equal variance, as for most statistical techniques (Fogelman-Soulie, 1991). Since the data is centred, the activation function must be symmetric, hence the use of the *tanh* function, or any other sigmoid. Data is scaled with small centred initial random weights so that it starts near zero (Kanellopoulos et al., 1991) thus avoiding saturation effects (Kanellopoulos *et al.*, 1993). The testing data is modified using the statistics calculated for the training file. Therefore, for two experiments which use the same testing file but different training data, the test file must be modified twice: once with the statistics of each training file.

When the pre-processing of the training and testing files is complete, they are divided into two files containing the training and testing data and two files containing the corresponding targets.

3.3.6 Accuracy of classification

It is important to distinguish between the classification accuracy for the training set and that for the testing set. The accuracy of classification of the training set provides a measure of how well the network can recognize patterns that it has already seen. On the other hand, the testing set accuracy is an indication of the generalisation capabilities of the network; in other words, how well the network recognises patterns it has never seen, which may not be exact copies of patterns in the training data. A good training accuracy does not necessarily indicate that there will be a good accuracy on unseen patterns. For

this reason, it is essential that the testing set be composed of pixels that are not in the training set. The testing set accuracy is also the measure which should be used to evaluate the performance of the network because it is the classification of unseen patterns that is of interest. Overall accuracy can be calculated by dividing the number of correctly classified pixels by the total number of pixels. It must be noted however, that even though overall accuracy may be quite high, some individual classes may not be well classified.

Sometimes, the testing set is used to tune the network (Yager, 1994). In other words, if the testing accuracy is unsatisfactory, parameters of the training procedure, for example learning rate, architecture and so on, are modified until the performance is satisfactory. In such a case, the testing file becomes part of the training set and an additional test set must be created (Le Cunn, 1987).

The main variable affecting classification is the training method, and not the classifier (Hepner *et al.*, 1990). This is the case for all classifiers, not only a neural network; the classifier learns to recognise classes on the information with which it is provided. If the information is erroneous, the results will be erroneous. For this reason, the poor quality of ground truth is a contributory factor to the poor performance of a network, and may be the cause of the lack of improvement in the performance of classifiers over the years (Wilkinson, 1997).

Network accuracy depends on network complexity but also on the composition of the two data sets, training and testing, as mentioned in the previous section. The amount of training data, the degree to which training data reflect true likelihood distributions and *a-priori* class probabilities affect the accuracy (Richard and Lippman, 1991). In addition, if the testing set does not approximately reproduce the class proportions of the training data, the accuracy will be biased in favour of some classes. If the training data is not an adequate representation of the actual data as provided by the image, the accuracy with regards to the testing set may not be impaired, if the test data contains similar proportions of classes, but the accuracy of classification with regards to the actual situation on the ground will be diminished or biased.

Training the network requires a balance between over-training and under-training. Over-training occurs when the network has learned the training set too well. The neural network has been forced to draw boundaries around outliers and overfitting occurs, resulting in a decrease in the testing accuracy (Rosin and Fierens, 1995). Undertraining occurs when the network has not been given enough time to learn a mapping between inputs and outputs.

The end of training occurs when a user defined number of iterations has been

completed or threshold error on the training data set has been reached. Neither of these parameters are usually set other than through trial and error. The best result obtained in a realistic amount of time has to be accepted, even though it may not be the optimum result, because of the slow rate of convergence (Wilkinson *et al.*, 1995a). If the number of iterations is the chosen criteria, the classification accuracy at the end of the training might still be unacceptably low. It is therefore recommended to train for a large number of iterations and to keep a record of the error. The number of iterations at which the error starts stabilizing can be used for subsequent experiments since it is unlikely that the error will diminish any further. The error is based on the training set and therefore only provides an indication of the level of accuracy of classification of the test set. However, if the training data is a representative sample of the testing data, the indication will be quite accurate.

The final level of accuracy which can be attained is usually totally unpredictable at the start of a training phase. The initial conditions for the training, that is, network architecture, weight initialisation and parameters, have an effect on the accuracy, although it is not clear whether it is significant (Wilkinson, 1997; Kanellopoulos and Wilkinson, 1997) or not (Paola and Schowengerdt, 1997). In any case, since standard multi-layer perceptrons have been shown to be capable of approximating any measurable function to any degree of accuracy, that is, they are universal approximators, any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden nodes or the lack of a deterministic relationship between input and target (Hornik *et al.*, 1989).

3.4 Modifications to the Original Software

When the thesis was started, the neural network classification software that was available had only been implemented for *pure* classification problems. In order to also carry out *soft* classification, it required modification. In addition, several software routines were written to ease the process of preparing data for input into the neural network, the most time consuming part of neural network classification. Tools for the analysis of neural network outputs were also developed. The original software was therefore incorporated into a menu driven package for classification. The chart in figure 3-12 illustrates the menu structure provided to help the user run the software. Although the software is an important part of this thesis, it is only summarised in the text but details are provided in appendix A which contains the source code and a description of each routine for those implemented by the author. The source code for the original software is

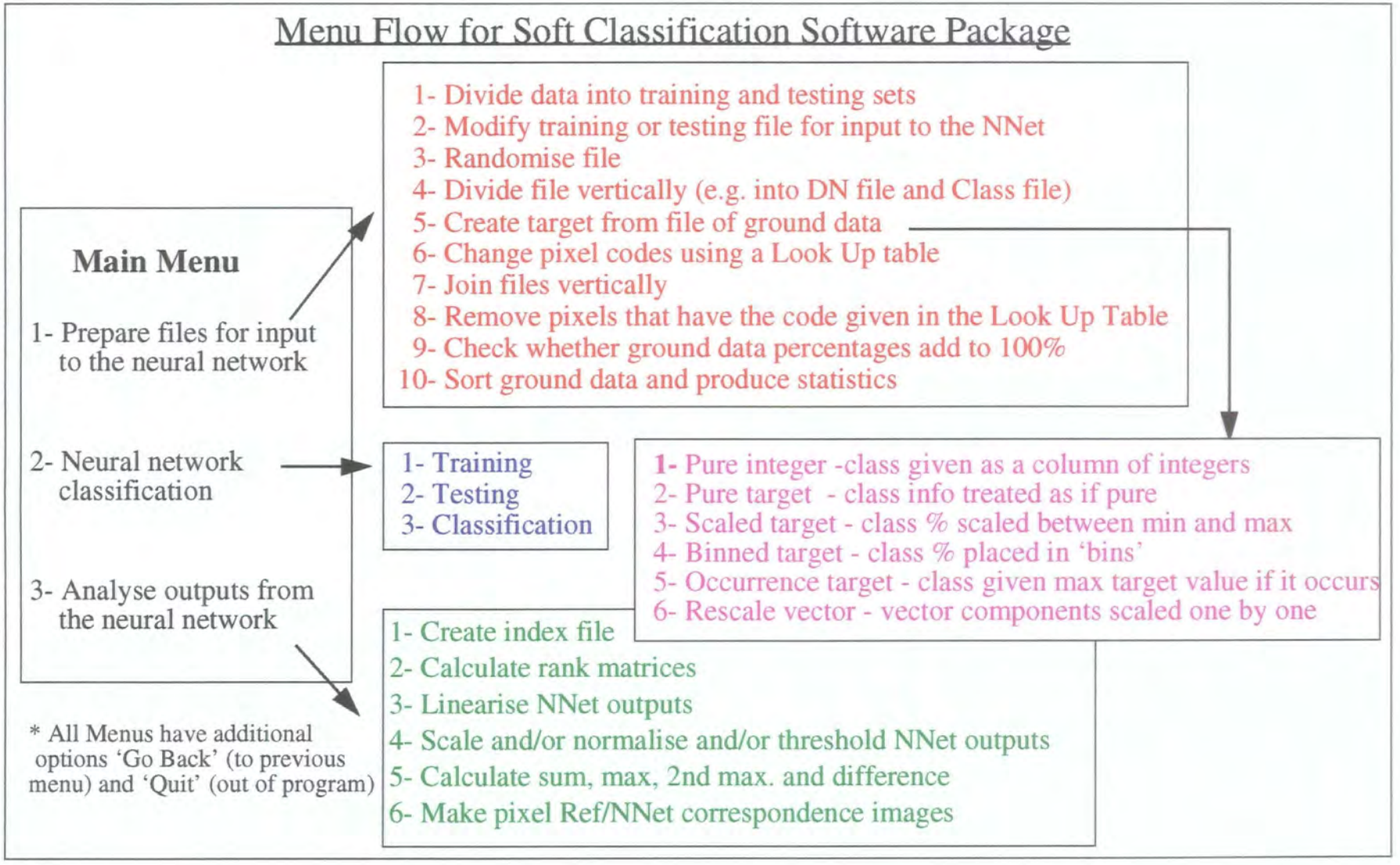


Figure 3-12. Chart showing the structure of the menus available to help the user run the *Soft* classification software

not provided; however, an outline of the modifications brought to the existing routines was provided in header comments which are included in appendix A.

The software is composed of three modules:

1- Preparation of files for input to the neural network

2- Neural network classification

3- Analysis of the outputs from the neural network.

The first option presents the user with a menu of operations to manipulate files and prepare data for input into the neural network. The second option provides a menu of neural network operations. The third option prints a menu of analysis tools for neural network outputs.

Within the 'Preparation' module, pixels with specific codes or ground data can be selected or removed, data files can be divided into training and testing sets, the files can be randomised, targets can be created, ground data can be checked, ground data statistics can be produced, and so on. Ground data are assumed to be in one of two formats. Either they consist of one column of integers indicating to which class a pixel belongs to (100% coverage is assumed), or they consist of integer vectors of ground data. The vectors of ground data can be of size 2 to $2m$, where m is the actual maximum number of mixture components found in the data set. The theoretical maximum number of components in a pixel in a data set is equal to the number of *pure* classes defined in the study. The actual maximum number of components of a pixel is usually lower. Pairs of vector elements are separated by spaces. In each pair of elements, the first integer indicates the percentage cover, the second number indicates the class. Thus, a mixture of [20% class 3 + 40% class 1 + 10% class 4 + 30% class 5] must be provided in this way: '20 3 40 1 10 4 30 5' (no quotes). Vectors of ground data can be sorted, for example in increasing order of cover or in increasing order of class number, but it is not a requirement. Coverage must add up to 100%.

The second module, 'Neural Network Classification' contains the modified network implementation. Within this module, classification takes place. The network can be trained, tested or an image can be classified. The original software assumed that training and testing data sets were provided to the network as one file for each data set: one for the training data and one for the testing data. Each row was expected to contain input data for one pixel: n columns of information such as modified channel DN values and in the last column, an integer indicating which class the pixel belonged to; 100% coverage was assumed. The integer class number was transformed into a target format

within the program and therefore without user intervention or even awareness. Targets were *pure*; the class to which the pixel belongs was assigned +1, the other classes were assigned -1. In the new software, mixed pixels can also be handled. Each data set, training and testing, is composed of two files. One file contains information such as modified channel DN values and the other file contains target representations of the ground data. Rows in the files correspond. That is, the pixel in row one of the information file is described by the target in row one of the target file and so forth. Targets have been created from the ground data and are not necessarily *pure* targets.

Other modifications concern the output of the neural network in the testing and classification stages. After testing, an output file can be created which provides the neural network output value for each node for each pixel of the testing file. This file is used to evaluate the potential of the neural network to identify sub-pixel components, using the analysis tools implemented in the third module of the program. At the classification stage, fraction images similar to those produced in linear unmixing models can be created. The image provided as input is divided into one image per class and the intensity of each pixel is the neural network output value for that class. These images are described in more detail in chapter VI. Details of the analysis tools which are provided in the third module are described in chapters VI and VII.

3.5 Summary of Chapter III

This chapter has provided an overview of the multi-layer perceptron used for the experiments described in this thesis. Training, testing and classification processes were outlined in section 3.1. They were then described in detail and the mathematical basis for the back-propagation algorithm was set out in section 3.2. The structure and parameters of the network were explained and their effect on neural network classification accuracy was discussed in section 3.3. Modifications to the original implementation of the network were summarised in section 3.4 which is complemented by appendix A. The next chapter describes the sites and test data collected with which experiments were carried out.

Chapter IV

REFERENCE DATA SITES AND DATA SETS

Whichever supervised classification method is used, the composition and quality of the training and testing data sets have an influence on classification accuracy. If a classifier is to perform satisfactorily, it must be given data which faithfully reflect the situation on the ground. There are two stages to creating training and testing data sets. The first is the collection of reference data; the second is the creation of data sets from the reference data. Both phases must be carried out taking particular care to ensure that the data sets produced will have characteristics that are well defined and understood. Classification accuracies can only be considered a true measure of the performance of the classifier if the properties of the training, testing and image data are known and can be used to assess the results.

In 1991, the EMAP Unit carried out an extensive ground campaign at a site in Portugal for its project concerned with providing a methodology for automatically updating CORINE (COoRdination of InformatioN on the Environment) landcover maps (Wilkinson *et al.*, 1991). Besides producing new maps of the chosen test units based on the CORINE nomenclature, the campaign allowed the EMAP unit to collect detailed ground information for homogeneous areas within the test units. Part of the information recorded included the composition of the parcels and the percentage of area covered by each class. This information was considered suitable for mixture analysis and is used in this thesis.

However, as work progressed, it became apparent that the data sets created from the reference data from Portugal were complex and that it may be helpful, for a better understanding of the network, to use a data set with few categories whose properties could be better controlled. Thus, a second site was chosen in Scotland. The site in Portugal, centred around the city of Lisbon, contains many different landcover classes (agricultural crops, trees, urban areas, water and so on) and many different types of mixtures. On the other hand, the site in Scotland consists only of plantations of coniferous trees where mixing occurs between trees and background grass. Obtaining percentage cover of pixels

by individual classes is particularly difficult within intimate mixtures. However, in the case of coniferous forests, structural attributes such as height and basal area of trees for example, have been found to be strongly correlated with reflectance values from forest stands (Cohen and Spies, 1992; Danson and Curran, 1993). In fact, the relationship is actually a function of the degree of canopy closure. Thus, it is hypothesised in this thesis, that measurements of height and basal area of forest stands can substitute for canopy cover and, therefore, for percentage cover of trees and background. In this way, the mixing between coniferous trees and background, and more particularly between Sitka Spruce trees and grass can be identified with a high degree of precision.

A detailed description of the method to acquire reference data and process them, and of the composition of data files is essential to the interpretation of results. This chapter provides a critical discussion of the data sets created from the reference data collected at the Portugal and Scotland sites. It is divided into several sections. The first section provides details of the satellite imagery for both sites. The second section outlines the methodology of the ground campaign carried out at the site in Portugal. This is followed by an analysis of the properties of each of the data sets created from the reference data collected. Then, the ground campaign at the Scotland site is described and, finally, the characteristics of the data sets created from the reference data are examined.

4.1 Location of Sites and Satellite Imagery

The geographical locations of the two sites in Portugal and Scotland are shown in figure 4-1. Photographs in figure 4-2 and figure 4-3 show the general characteristics of the landscape at each site. As can be seen, the two sites are quite different.

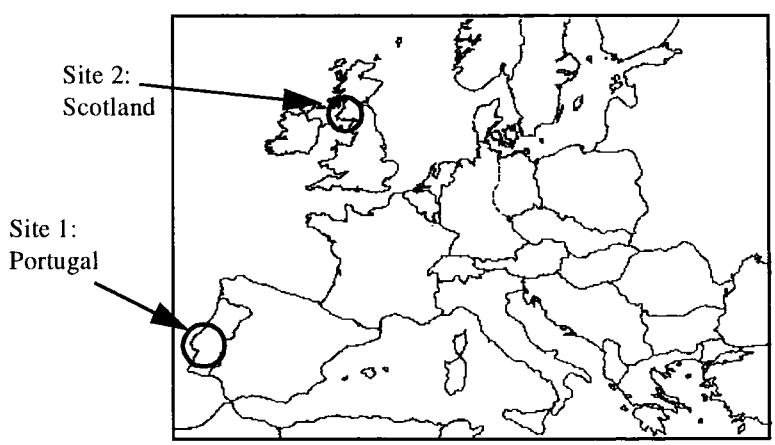


Figure 4-1. Geographical location of sites from which reference data was collected

The satellite imagery was acquired by the Thematic Mapper (TM) sensor of the Landsat 5 satellite. The images used in the thesis are six band quarter scenes. The thermal

Figure 4-2. Photograph showing the typical landscape for the site in Portugal

Figure 4-3. Photograph showing the typical landscape for the site in Scotland

channel of the TM sensor (band 6) was not used because of its lower spatial resolution of approximately $120m^2$ compared to the other channels' resolution of approximately $30m^2$. The images have very little haze or cloud cover and are of good visual and radiometric quality. Atmospheric correction was not deemed necessary as no temporal analysis was carried out and atmospheric effects over a scene (and moreover over a quarter scene) appear almost constant (Chikkara, 1984).

The main characteristics of the images are listed in table 4-1. One image was used for Portugal which was acquired in geocoded format, details of which were not available. In the absence of the author having access to the geocoding information, the locational accuracy of polygons was assessed visually. Two images were used for Scotland from which a small area was extracted. The images were geometrically corrected (by a member of the same research group as the author at Durham University) using 25 ground control points, and georectified to the UK National Grid. The root mean square error of rectification was just over half a pixel. Figure 4-4 illustrates the image from Portugal and the 1995 image from Scotland and a subscene for the latter, showing the study area.

Site	Date	Path / Row	Upper Left	Lower Right	Geocoding
Portugal	June 24th 1991	Path = 20	x = 471,450	x = 543,450	to Universal Transverse Mercator (UTM)
		Row = 33	y = 4,327,800	y = 4,255,800	
Scotland	June 21st 1995	Path = 205	x = 175,297	x = 303,427	to UK National Grid
		Row = 22	y = 620,898	y = 506,268	
	July 11th 1989	Path = 205	x = 186,183	x = 295,713	
		Row = 22	y = 617,673	y = 512,013	

Table 4-1. Details of the satellite images

4.2 Collection of Reference Data for Portugal

The ground campaign was carried out by staff from the EMAP unit and from the Universidad de Lisbõa (Portugal) in June 1991, contemporaneously to the acquisition of the satellite image. A total of 12 sites of $3x3km^2$ were visited, some twice, covering an area of approximately $60x60km^2$ in and around Lisbon. 736 polygons were mapped. Figure 4-2 shows the typical landscape of the study area. The guidelines issued for the field survey (Wilkinson *et al.*, 1994) recommended a minimum mapping area of 4ha (200m*200m). The field staff were asked to delineate homogeneous parcels on maps (1:25,000) and on aerial photographs (1:33,000) which were later digitised. Photographs were unfortunately only available for part of the test units at the time of the survey (Wilkinson and Folving, 1991). Homogeneous landcover types are defined as consisting

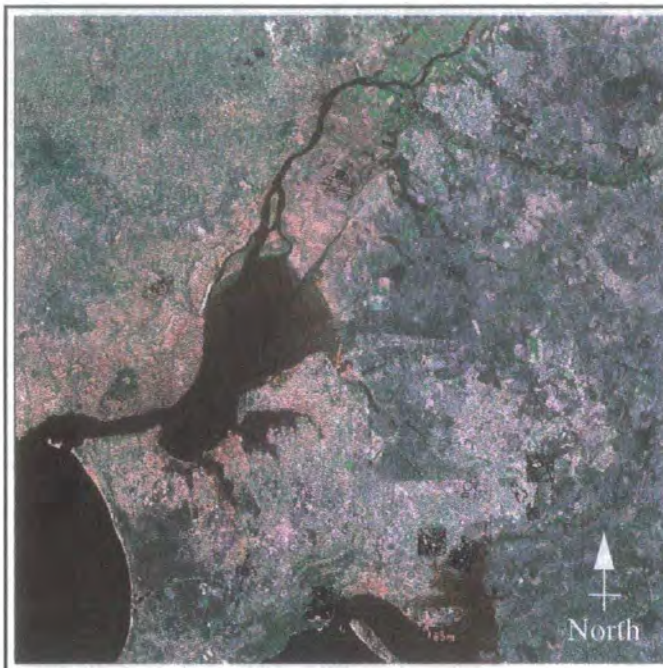


Image of Portugal site (1991)

RGB = Bands 3, 4, 5
black = polygons from which ground reference data extracted

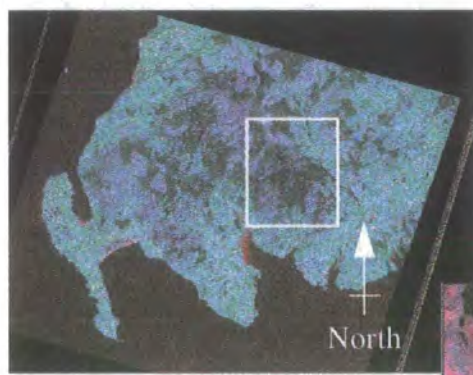


Image of Scotland site (1995)

RGB = Bands 3,4,5
white box = location from which subscene extracted

Subscene from 1995 Scotland image

RGB = Bands 3, 4, 5
white = polygons from which ground reference data extracted

Figure 4-4. Sample satellite images and location of reference data collection sites

of one *pure* class, or consisting of an intimate mixture of *pure* classes where the physical distance between two classes is much less than one pixel and regular, for example an orchard. In this way, it can be assumed that any pixel extracted from the polygon will have the same composition as the polygon. A form was provided to record the composition of each parcel, a copy of which can be found in appendix B, section B.1. The information recorded for each polygon could include up to three vegetation species with their respective heights and percentage cover of the parcel; water content of surface (wet/dry); colour of surface not covered by vegetation; and composition of the surface not covered by vegetation. Percentage cover was estimated visually. Over seventy different landcover classes were recorded. The information was entered into an Oracle database and Arc/Info GIS.

From this reference data, three data sets were created containing sixteen, fifteen and seven *pure* classes. Throughout the thesis, these data sets will be referred to as 'Portugal sixteen class', 'Portugal fifteen class' and 'Portugal seven class' respectively.

4.3 'Portugal Sixteen Class' Data Set

4.3.1 Methodology for creating the data set

A data set was made available to the author which had been created from the reference data in the following manner.

1- From the seventy or more reference classes reported in the survey, sixteen *pure* categories (listed in table 4-2) were defined to represent the landcover of the image.

2- (a) Where possible, for each class, polygons covered 100% by the class were selected.

(b) For some classes, in particular those which rarely occupy 100% of a polygon, the data were supplemented by selecting polygons which had the next highest coverage by the class. Examples of such classes include **Bare Soil**, **Fresh Water**, **Maize**, **Aquatic Plants**, **Weeds** and **Grass**.

(c) For classes which never occupy 100% of a polygon, pixel data were extracted from polygons with a high coverage by the class but not complete (100%) occupancy. Examples of such classes include: **Tiled/Concrete**, **Barley**, **Vines**, **Garrigue**, **Deciduous Forest** and **Coniferous Forest**. Regardless of the actual composition of the polygon on the ground, it was considered to be occupied 100% by the dominant class and assigned that class number. This is a common

method of producing training data for conventional classification when *pure* polygons are not available.

3- A total of 12,505 pixel vectors were extracted from the image consisting of, for each pixel, the DN value for each of the six image bands and the class number of the polygon from which the pixel was extracted.

Id.	Class Name	Pixels	Id.	Class Name	Pixels
1	Tiled / Concrete	314	**9**	Maize	282
2	Sand	172	**10**	Aquatic Plants	501
3	Bare Soil	868	**11**	Vineyards	1,046
4	Sea Water	368	**12**	Weeds	1,141
5	Fresh Water	277	**13**	Garrigue	1,465
6	Estuary / Lagoon	721	**14**	Grasslands	3,331
7	Wheat	423	**15**	Deciduous Forest	828
8	Barley	311	**16**	Coniferous Forest	457
Total number of pixels					12,505

Table 4-2. Composition of the 'Portugal sixteen class' data set

4.3.2 Analysis of the data

The 'Portugal sixteen class' data set contains no mixture information which is typical of data sets created for conventional *pure* classification problems. As shown in table 4-2, the number of pixels in each class differs. As discussed in chapter III, section 3.3.5, the number of pixels within each class may provide the network with an *a-priori* weighting of the importance of that class and the testing file should therefore be representative of the distribution in the image; only then are the classification accuracies representative. The number of pixels per class for this data set were chosen partly on this basis (Wilkinson, personal communication).

Spectral analysis

An overview of the spectral separability of classes can be estimated by comparing their spectral signatures. Generally, the spectral graph of a class shows the mean value and standard deviation of each band. However, box plots show a number of other characteristics of the data beyond the simple mean and are used in this thesis. In a single box plot such as that represented in figure 4-5, the median of the data values is shown by a white line in the box. The spread of values is shown by the height of the box. The top of the box lies at the 75^{th} percentile, the position in the data distribution below which 75% of

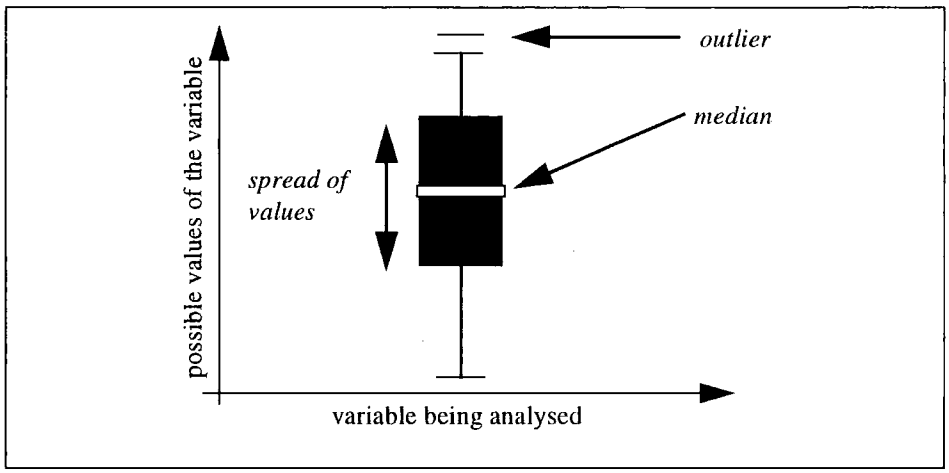


Figure 4-5. Illustration of a box plot

the data lie, while the bottom of the box lies at the 25th percentile. Thus 50% of all values lie within the box whose vertical height is referred to as the Inter Quartile Distance (IQD). Asymmetry in the data is shown by the position of the white line, the median, with respect to the vertical dimensions of the box. When plotting spectra, horizontal dimensions are meaningless. However, if the same variable is being compared, for example band 1 for several classes, then the width can be made to represent the number of samples which are present in the distribution. The vertical lines which extend beyond the boundaries of the box extend to the extreme values of the data or a distance of 1.5 *IQD from the centre, whichever is less. Data points which fall outside the vertical lines are outliers. Box plots of spectral signatures can be used to identify:

- classes which have similar spectral signatures and may be expected to be difficult to distinguish,
- the spread of values of each class, which provides an indication of the variability of spectral signatures within the class,
- the number of outliers of each class, which shows how well the class is represented by statistics such as the mean and the variance.

Figure 4-6 shows the spectra of the sixteen classes listed above, some of which are discussed in more detail below (n = number of pixels).

The classes with the least spread and outliers are **Sea Water** and **Estuary**, shown in figure 4-7. Water is usually a well defined class with little variability so this is not surprising. On the other hand, **Sea Water** and **Estuary** classes share very similar spectral signatures and may be expected not to be well differentiated. **Fresh Water** and **Coniferous Forest**, shown in figure 4-8, appear to have the most outliers suggesting less

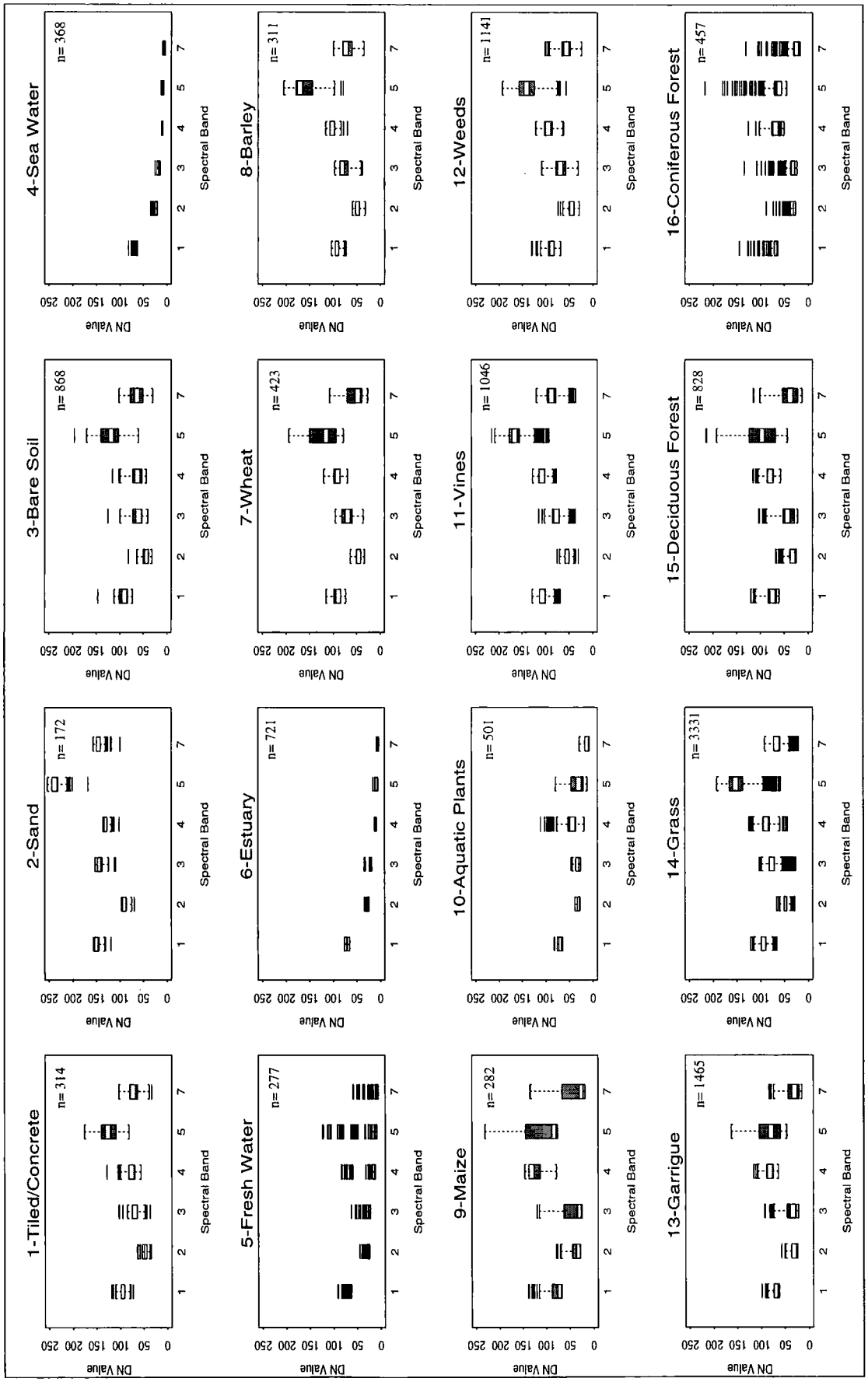


Figure 4-6. Spectra of the 'Portugal sixteen class' data set

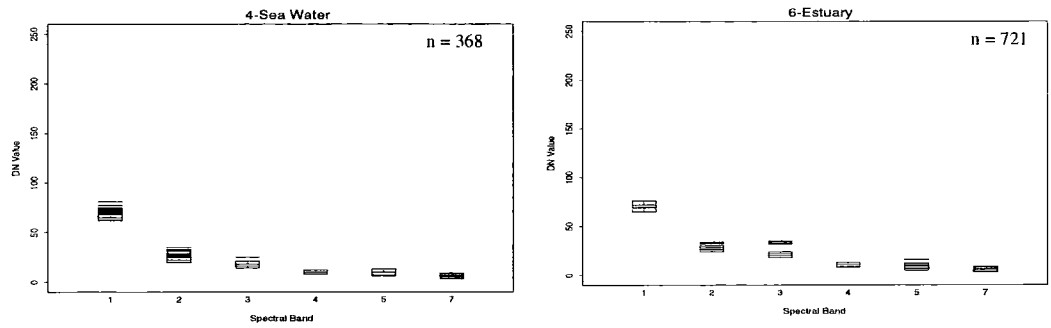


Figure 4-7. Spectra of the Sea Water class and the Estuary class of the 'Portugal sixteen class' data set

well defined classes and consequently more confusion with other classes at the classification stage. The reason for outliers in the **Fresh Water** class may be the presence of vegetation within these pixels. This is suggested by the spectral signature of some of the pixels which is typical of vegetation such as **Grass**. The cause of outliers in the **Coniferous Forest** class is not clear. Both of these classes have a very small spread for 50% of their values which means that there is more likelihood of there being outliers. For example, if a class is defined by spectral values which have quite a large spread, as for example **Wheat**, then the likelihood of there being outliers for that class is smaller than for classes defined by very precise, and therefore small spread, values. This is because 1.5*(IQD), defined above as being the distance beyond which points are considered to be outliers, is larger for poorly defined classes than for precisely defined classes.

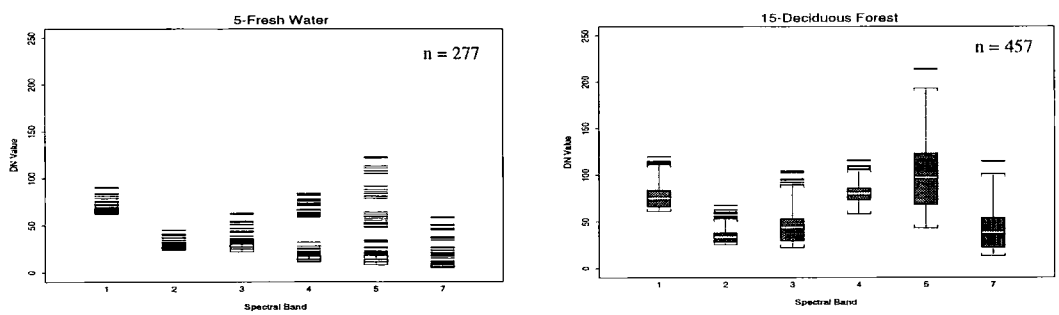


Figure 4-8. Spectra of the Fresh Water class and the Coniferous Forest class of the 'Portugal sixteen class' data set

The spectral signature for the non-water classes share a similar pattern: decrease in the signature between bands 1 and 2, increase between bands 2 and 5, decrease between bands 5 and 6. The position of the medians vary, as do the spread and the number of outliers. For example, the mean value in all the bands for **Sand** are higher than for any of the other classes. This makes it likely that the **Sand** class will be separable from the others. On the other hand, figure 4-9 shows that there is little difference between the **Tiled/ Concrete** signals and the **Bare Soil** signal for example.

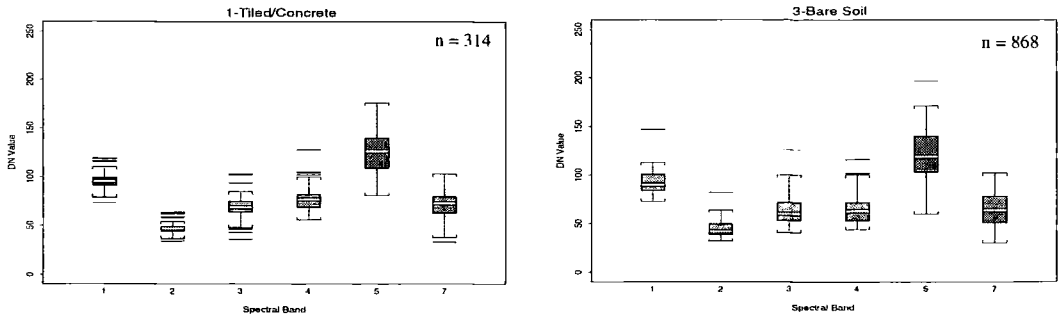


Figure 4-9. Spectra of the Tiled/Concrete class and the Bare Soil class of the 'Portugal sixteen class' data set

In summary, the box plots are useful prior to classification to determine expected results and after classification to examine the reasons why classes may be confused. Indices which provide a mathematical representation of the difference between classes based on some form of distance measurement could also be used to quantify the separability of classes. However, these techniques usually produce a single number, which is to be taken as an indication of the separability of classes, which does not reveal any information on possible mixing between classes.

Quantile-quantile plots

As mentioned in chapter II, section 2.3, the maximum likelihood classifier assumes that classes are normally distributed whereas the neural network makes no such assumption. Quantile-quantile plots are used to compare the distributions of two sets of data and can be used here to test the actual distribution of the data against the assumption of normally distributed data. In the plots, the percentiles of the data are compared against the same percentiles for, in this case, a standard normal; that is a normal with a mean of zero and a standard deviation of one. Even though one data set consists of discrete values (integer DNs) and the other consists of continuous values (the standard normal), provided there are enough points covering the range of values, this comparison is valid. On these quantile-quantile plots, if the data lies on the line, the actual data can be considered to be normally distributed. In conjunction with the spectral plots, the quantile plots can show whether there is a correlation between class definition and normal distribution of the data. Quantile plots were drawn for each band of each class. A sample of the plots are shown in figure 4-10 to figure 4-12. All quantile plots can be found in appendix B, section B.2.

Figure 4-10.a is the quantile plot for **Sea Water**, band 2. It shows that even though a class may be well defined spectrally as shown in figure 4-6.4, this does not imply a

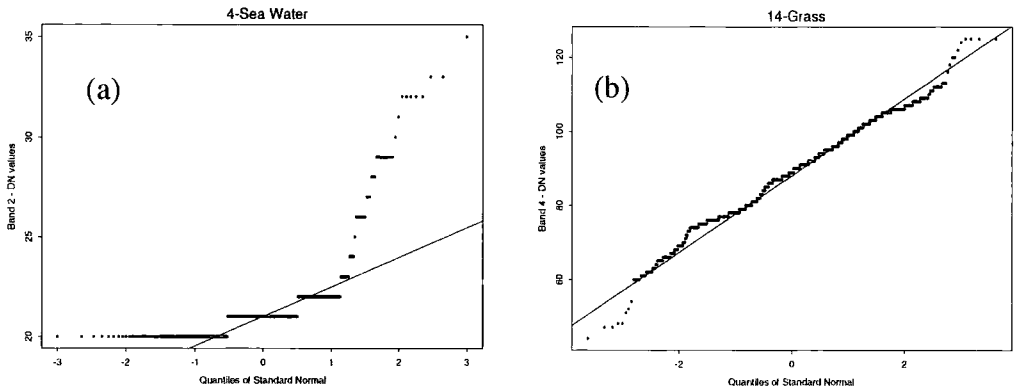


Figure 4-10. Quantile-quantile plots for band 2 of the Sea Water class (a) and band 4 of the Grass class (b) of the 'Portugal sixteen class' data set

normal distribution. On the other hand, a class with more variance in its spectra, such as **Grass**, may show approximately normal distribution in one of its bands, in this case band 4 (figure 4-10.b). It seems therefore that there is no relation between class definition, and more particularly, the spread or not of values for each spectral band, and normal distribution. The step-like aspect of the **Sea Water** figure compared to the **Grass** figure is because of the scale of its ***y-axis***. The shape of the plot is not a function of band number or class. Different classes may have similar distributions as illustrated by graphs for **Coniferous Forest** and **Fresh Water** (figure 4-11).

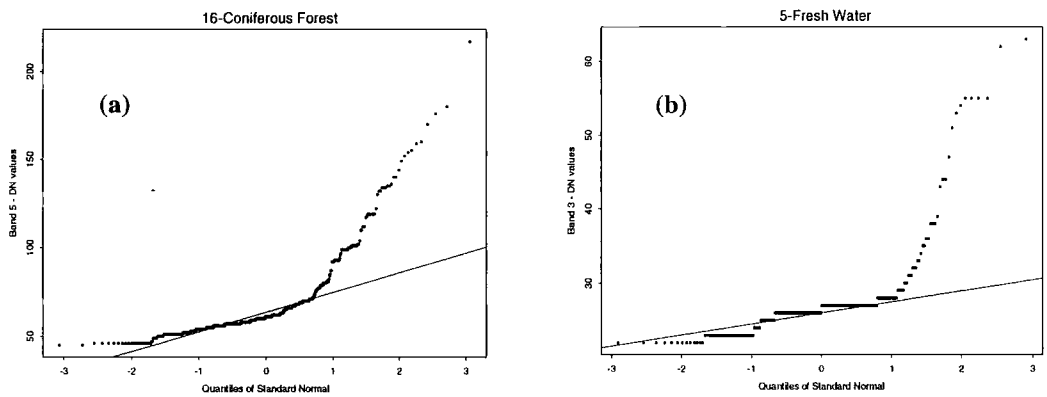


Figure 4-11. Quantile-quantile plots for band 5 of the Sea Water class (a) and band 3 of the Fresh Water class (b) of the 'Portugal sixteen class' data set

The same class may show normality in one of its bands but not another. Figure 4-12 shows that band 4 of the **Vines** class seems to be approximately normally distributed whereas the data for band 5 is not. The quantile-quantile plots show in which direction data is skewed. For example, in the quantile plot for band 5 of the **Vines** class, the actual data has a larger tail in its lower values than the standard normal and is therefore negatively skewed. The quantile-quantile plots confirm that data is not always normally

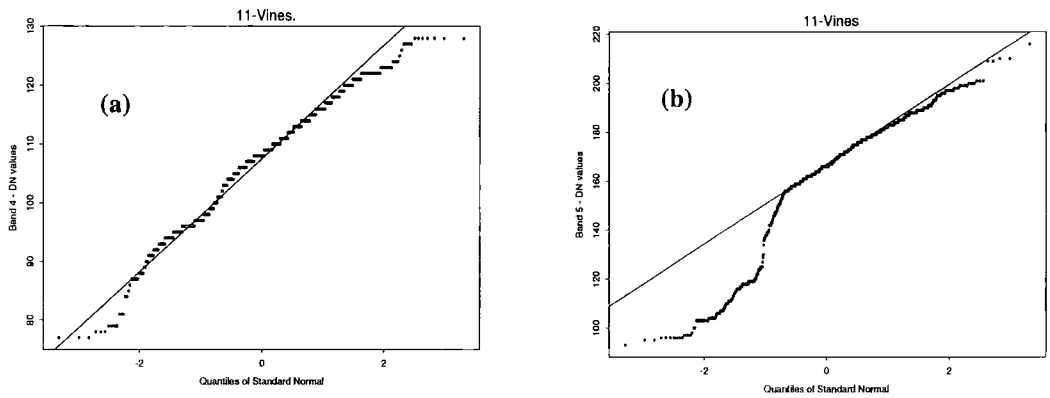


Figure 4-12. Quantile-quantile plots for band 4 of the Vines class (a) and band 5 of the Vines class (b) of the 'Portugal sixteen class' data set

distributed and that a technique such as maximum likelihood, which specifically assumes normality of the data may not be appropriate.

The 'Portugal sixteen class' data set did not take advantage of the mixture information available for each polygon. Indeed, the aim of the process of extracting training data was to collect pixels which were as *pure* as possible. Location information for the pixels was not available. Therefore, a new data set was created from the reference data for the purpose of analysing mixtures.

4.4 'Portugal Fifteen Class' Data Set

The data set to be created required detailed mixture information. A mixture is defined both by its class composition and the percentage cover by each component; thus a mixture of [60% class 1 + 40% class 4] is considered to be a different mixture from [50% class 1 + 50% class 4].

The creation of this data set from the reference data required five steps:

1- Correction of mistakes

2- Removal of ambiguous polygons

3- Reduction of the number of landcover types

4- Buffering

5- Simplification

In the work presented in this thesis, the aim of the experiments was not the classification of images but rather an evaluation of the potential of the classifier. For this reason, the training and testing data sets were created to be representative of each other but not necessarily of the image. In fact, making training data sets that are representative of the

image to be classified is particularly difficult for *soft* classification problems as it implies a high degree of *a-priori* knowledge about the landcovers in the scene.

4.4.1 Methodology for creating the data set

1 - Correction of mistakes

Since reference data is considered to be true and correct, mistakes must be minimised. Checking the reference data revealed a number of typographical mistakes and inconsistencies of class names which were corrected. In addition, the percentage coverage per class of some polygons did not sum to 100%. It is difficult to decide how to correct this; an arbitrary decision was taken to always add to the smallest percentage or to subtract from the largest. At this stage it was decided that no other modification could be made and this was considered the final and true ground information from which a data set could be created. There were 797 polygons.

2 - Removal of ambiguous polygons

Polygons with the following characteristics were removed from the data base:

- whose landcover was either unique or ambiguous (for example: bregner (danish grass), salt, sun flower, wild oats, mixed, corn? (sic), pine-oak (no relative percentage coverage provided), trees (no species indication)),
- which included the comment 'non-homogeneous',
- for which the original survey form was missing and could not therefore be checked,
- for which there was no composition information,
- whose percentage composition was arbitrarily corrected in the previous step

3 - Reduction of the number of landcover types

Over seventy landcover classes had been recorded. Besides being unmanageable, many of these can be expected to have similar spectral responses. The landcover classes were therefore divided into 15 groups based on theme or expected spectral signatures.

- The class name **Fruit Trees** grouped together vegetation types which included: apples, apricots, cherries, figs, fruit trees, olives, oranges, peaches, pears, plums and prunes.
- The class name **Horticulture** grouped together vegetation types which

included: asparagus, beans, carrots, melon, mixed horticulture, potatoes, strawberries and tomatoes.

- The class name **Cereals** grouped together vegetation types which included: barley, corn, maize, oats, wheat.
- The class name **Broadleaf** grouped together vegetation types listed as: cedar, eucalyptus, oak. Broadleaf is used in preference to deciduous because some of the tree types such as the eucalyptus trees do not actually shed their leaves.
- The class name **Grass** grouped together vegetation types listed as: grass or weeds.
- The class name **Coniferous** replaced the vegetation type listed as: pine.
- The class name **Marsh** grouped together vegetation types which included reeds or rice.
- The class name **Urban** grouped together non-vegetation types which included: asphalt, concrete, construction material, metal, tile, urban, gardens, plastic.
- The class name **Stones** grouped together non-vegetation types which included: dirt road, quarry, stones, rock, rubble, gravel, non-asphalt roads.
- **Shrubs, Stubble, Vines, Bare Soil, Sand and Water** were kept as class names.

At this point there were 15 landcover classes and 303 different mixtures. Each of the fifteen class is considered to be a *pure* class even though it may be composed from several original landcover classes.

4 - Buffering

Polygons were then buffered within the GIS by 30m (1 pixel) to eliminate the problems which arise from pixels lying on boundaries. As figure 4-13 shows, border pixels may be assigned to no class or to the wrong class because of location errors between lines and points in a GIS and because of the accuracy of the geocoding and georeferencing processes, usually half a pixel (Thomas *et al.*,. 1987). The green pixels in figure 4-13.a could be assigned to no class. The orange pixel in figure 4-13.a could be assigned to class **B**, even though it actually belongs to class **A**. Buffering removes these problem pixels as shown in figure 4-13.b, so that the remaining pixels can confidently be said to belong to the polygons they are identified with and therefore to that class composition. When

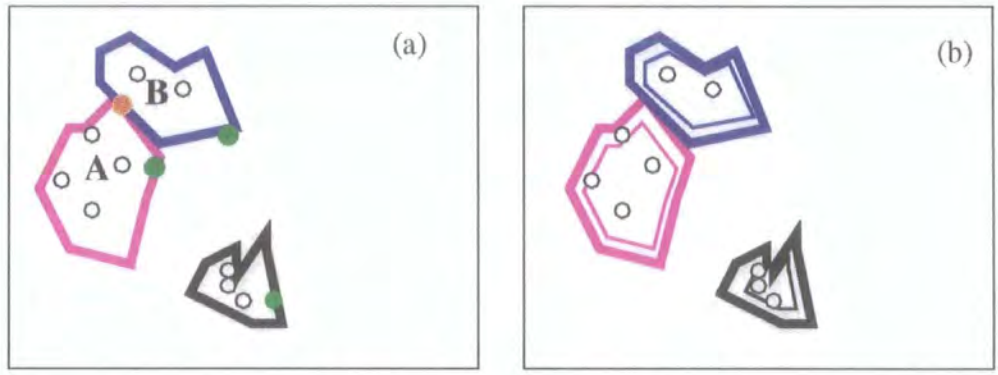


Figure 4-13. Illustration of the problem of boundary pixels (a) and buffered polygons (b)

buffering was applied, some polygons automatically disappeared as they became too small to be represented. In addition to these, polygons with an area smaller than 3600m^2 were also removed. For a square polygon this would correspond to a size smaller than 2x2 pixels. Figure 4-14 shows the results of this set of operations on a test unit. At this stage, 483 polygons remained of 233 different mixtures, and 42,589 pixels were extracted.

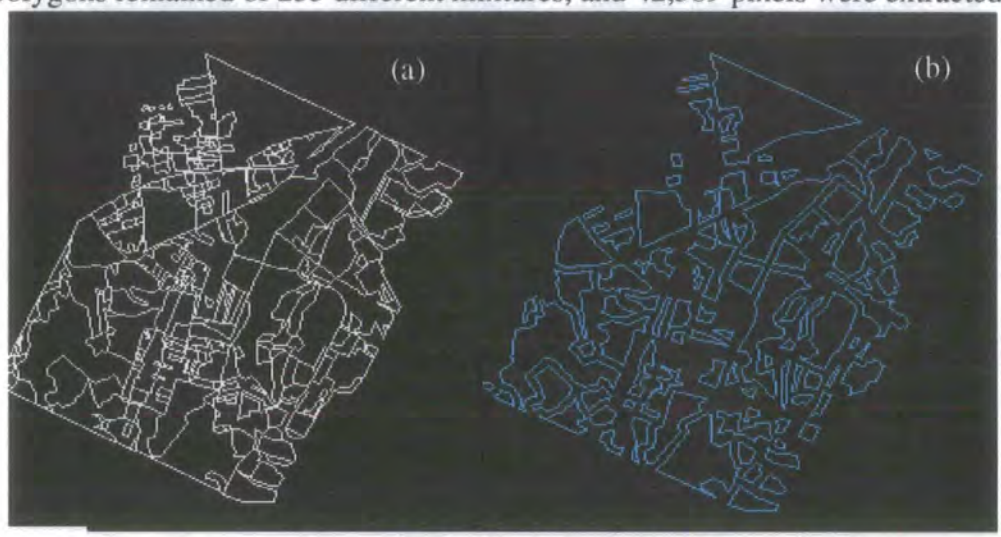


Figure 4-14. Digitized boundaries of (a) polygons and (b) polygons after buffering for one of the sites surveyed in Portugal

5 - Simplifying

The data set was further reduced by:

- removing polygons with mixtures containing a component less than 10% (92 mixture types e.g. [40% class 1 + 2% class 3 + 58% class 11]),
- removing polygons with mixture components which were not multiples of 10 (e.g. [25% class 10 + 75% class 14]),
- removing polygons with [50%-50%] compositions (no dominant class).

At this point, 108 mixture types remained. The data set contains 27,651 pixels extracted from the image. The list the 108 of cover types which make up the fifteen class data set are listed in appendix B, section B.3. The list of cover types removed in steps 4 and 5 above are listed in appendix B, section B.4. The accuracy of classification of the test data set assuming that all pixels are *pure* was about 70% for this data set whereas it was about 50% before being simplified.

4.4.2 Analysis of the data

Spectral analysis

The analysis of *soft* classification training and testing data, that is mixed pixel information, is very complex unless there are few classes and mixtures. For example, in this data set there are 108 different mixtures which makes it impossible to study individual spectral signatures for each composition. Furthermore, not all of the fifteen *pure* classes occur as 100% cover of a polygon. One of the unfortunate consequences of simplifying the data is the removal of polygons whose coverage is almost *pure*; for example, whereas class 3 covered 90% of a polygon in the mixture [6% class 1 + 90% class3 + 3% class 6 + 1% class 10], after simplification, the only occurrence of class 3, **Stones,** was in the mixture [10% class 3 + 90% class 12]. The maximum occupancy for some classes such as **Vines** and **Fruit Tree** was only 50% or 60%. Therefore, for each of the fifteen classes, an example composition was chosen where the percentage cover of the class is maximised; they may not necessarily be dominant in terms of percentage cover for the reasons given above. The examples are listed in table 4-3 and their spectra are shown as box plots in figure 4-8.

By plotting the spectra of the examples listed in table 4-3, a similar analysis to that provided for the 'Portugal sixteen class' data set can be carried out although it is restricted to the examples that have been chosen. Figure 4-15 shows the spectra for the 'Portugal fifteen class' data set. Numbers of pixels per class vary greatly here, but these are not the only pixels in the data set for these classes.

- **Stones**, **Stubble**, and **Fruit Trees** have quite well defined signatures with little spread or outliers.
- The **Stones** class has a fairly distinct signature which, strangely it is thought at first, is more similar to the **Shrubs** class than to the **Urban** class for example. However, considering that **Stones** only exists as part of a mixture with 90% **Shrubs**, it is not surprising.

Id.	Class Name	Example Composition of Polygon with Maximum Percentage Occurrence of the Class	Number of Pixels
1	Bare Soil	100% 1	663
2	Sand	100% 2	69
3	Stones	10% 3 +90% 12	93
4	Water	100% 4	798
5	Marsh	90% 5 + 10% 1; 90% 5 + 10% 4	108
6	Urban	100% 6	362
7	Stubble	100% 7	140
8	Cereals	100% 8	249
9	Horticulture	70% 9 + 30% 1	39
10	Grass	100% 10	1,899
11	Vines	60% 11 + 10% 10 + 30% 1; 60% 11 + 40% 10	1,342
12	Shrubs	90% 12 + 10% 3; 90% 12 + 10% 10	299
13	Fruit Trees	20% 1 + 30% 10 + 50% 13	9
14	Broadleaf	90% 14 + 10% 10	10
15	Coniferous	90% 15 + 10% 12	5

Table 4-3. List of *pure* classes of the 'Portugal fifteen class' data set and example mixture compositions

- The **Stubble** class's spectral signature overlaps the **Bare Soil** class which makes it likely that the two will be confused at the classification stage.
- **Fruit Trees** have a signature which is similar to **Shrubs** but with slightly higher DN numbers for all bands except band 4.
- **Horticulture** is a slightly less well defined class, with a small number of outliers, and its spectral signature is similar to that for **Cereals.**
- For the classes which were also defined in the 'Portugal sixteen class' data set, there are some differences. For example, **Bare Soil** has a larger spread of values in the 'Portugal fifteen class' data set than the 'Portugal sixteen class' data set, as does the **Sand** class, even though they both have fewer pixels than in the previous data set. The **Vines** class also has a larger spread of values but it also has more pixels.

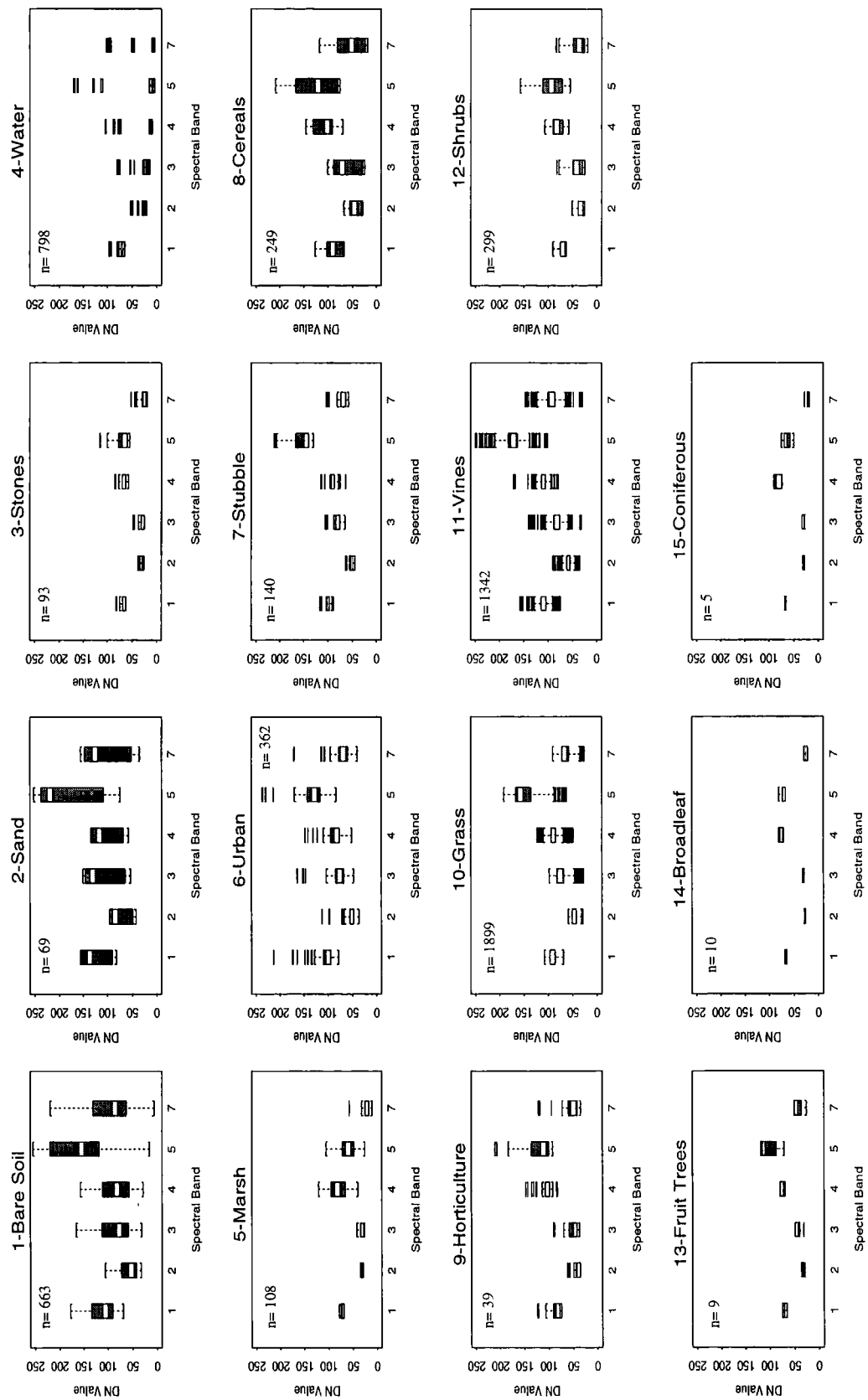


Figure 4-15. Spectra of the 'Portugal fifteen class' data set (n = number of pixels)

- On the other hand, the **Broadleaf** and **Coniferous** forest classes have more precisely defined signatures in the 'Portugal fifteen class' data set than in the 'Portugal sixteen class' data set. The outliers present in the sixteen class data set have been removed. This is not surprising considering the number of pixels plotted here (10 and 5 respectively). There is very little difference spectrally between these two classes, and between the two classes and the spectral signature for **Shrubs**. This leads to the expectation that these classes may be confused at the classification stage.
- The **Shrubs** class has a very similar signature to its equivalent **Garrigue** in the sixteen class data set.
- On the other hand, the **Marsh** class and its equivalent, the **Aquatic Plants** class in the sixteen class data set, surprisingly do not have similar signatures. The **Marsh** class has higher DN values in bands 4, 5 and 6. It is not clear why this is so.
- The **Urban** class of the fifteen class data set is less well defined than the **Tiled/Concrete** class of the sixteen class data set but their definition was also different, the original classes that make up the **Urban** class being more varied.

Finally, there are three classes in the 'Portugal fifteen class' data set which group together classes from the 'Portugal sixteen class' data set:

- **Water**, **Cereals** and **Grass**. Not surprisingly, these classes have more spread and more outliers than the single classes from the sixteen class data set since they group two or more signatures together.

From the spectral analysis, it could be decided to remove or redefine some classes such as the **Stones** class for example. This was not done here as the data were not intended for image classification and a spread of quality in the classes was felt to provide a better overview of the potential of the neural network classifier.

Composition matrix

The composition of the data set is particularly difficult to summarise because of the large number of different mixtures. One type of summary table is presented in table 4-4. Mixtures are read as follows.

Mixtures are ordered on their dominant class, the class which individually covers the largest percentage area. In the column titled 'Class Name of Dominant Class', if the

Id.	Class Name of Dominant class	Class Name of Secondary Class in Two Component Mixture	Class Names of Secondary/ Third Classes in Three Component Mixtures	Class Names of Second, Third, Fourth Classes in Four Component Mixtures
1	Bare Soil	6- Urban 7- Stubble 8- Cereals 9- Horticulture 10- Grass 11- Vines 13- Fruit Trees	**10 - Grass / 6 - Urban** **10 - Grass** / 11 - Vines **10 - Grass** / 14 - Broadleaf 11 - Vines / **10 - Grass** 11 - Vines / 13 - Fruit Trees 12 - Shrubs / 13 - Fruit Trees 13 - Fruit Trees / **10 - Grass** 13 - Fruit Trees / 11 - Vines 13 - Fruit Trees / 12 - Shrubs 14 - Broadleaf / **10 - Grass** 15 - Coniferous / 12 - Shrubs	**10 - Grass** / 13 - Fruit Trees / 12 - Shrubs 13 - Fruit Trees / **10 - Grass** / 12 - Shrubs
2	Sand			
3	Stones			
4	Water	1 - Bare Soil 5- Marsh		
5	Marsh	1- Bare Soil 4 - Water 10 - Grass		
6	Urban	10 - Grass 12 - Shrubs		
7	Stubble	1- Bare Soil		
8	Cereals	1- Bare Soil 10 - Grass 14 - Broadleaf		
9	Horticulture	1 - Bare Soil	1 - Bare Soil / **6 - Urban** 1 - Bare Soil / **10 - Grass**	
10	Grass	1 - Bare Soil 4 - Water 5 - Marsh 8 - Cereals 12 - Shrubs 13 - Fruit Trees 14 - Broadleaf	1 - Bare Soil / 11 - Vines 11 - Vines / 1 - Bare Soil 12 - Shrubs / 15 - Coniferous 13 - Fruit Trees / 1 - Bare Soil 14 - Broadleaf / 1 - Bare Soil 15 - Coniferous / 1 - Bare Soil	1 - Bare Soil / 11 - Vines / 13 - Fruit Trees 1 - Bare Soil / 13 - Fruit Trees / 11 - Vines 13 - Fruit Trees / 1 - Bare Soil / 12 - Shrubs 1 - Bare Soil / 13 - Fruit Trees / 12 - Shrubs
11	Vines	1 - Bare Soil	1 - Bare Soil / **10 - Grass**	1 - Bare Soil / **10 - Grass** / 13 - Fruit Trees
12	Shrubs	3 - Stones 6 - Urban 10 - Grass 15 - Coniferous	1 - Bare Soil / 15 - Coniferous **10 - Grass** / 15 - Coniferous 14 - Broadleaf / **10 - Grass** 14 - Broadleaf / 1 - Bare Soil	1 - Bare Soil / 14 - Broadleaf / 15 - Coniferous 1 - Bare Soil / 15 - Coniferous / 14 - Broadleaf 15 - Coniferous / 1 - Bare Soil / 14 - Broadleaf 15 - Coniferous / 14 - Broadleaf / 1 - Bare Soil
13	Fruit Trees	1- Bare Soil	**10 - Grass** / 1 - Bare Soil	1 - Bare Soil / **10 - Grass** / 12 - Shrubs **10 - Grass** / 1 - Bare Soil / 12 - Shrubs
14	Broadleaf	10 - Grass 12 - Shrubs	**10 - Grass** / 1 - Bare Soil 12 - Shrubs / 1 - Bare Soil 12 - Shrubs / **10 - Grass** 15 - Coniferous / 12 - Shrubs	
15	Coniferous	10 - Grass 12 - Shrubs	**10 - Grass** / 1 - Bare Soil 12 - Shrubs / 1 - Bare Soil 14 - Broadleaf / 12 - Shrubs	12 - Shrubs / 1 - Bare Soil / 14 - Broadleaf 12 - Shrubs / 14 - Broadleaf / 1 - Bare Soil

Table 4-4. Summary of the composition of mixtures in the 'Portugal fifteen class' data set

class name is underlined, it exits as a *pure* polygon, covered 100% by that class. For each row, the second column lists the secondary classes in two class mixtures with each of the first classes listed in the first column. The mixtures are organised so that they only appear once in the row of the dominant class. Thus a **Grass - Bare Soil** mixture, where **Grass** is the dominant class will be listed under class 10, **Grass**, but not under class 1, **Bare Soil**. Furthermore, classes are listed in decreasing order of importance. Thus a three class mixture **Bare Soil - Grass - Urban** is not the same as a three class mixture **Bare Soil - Urban - Grass**. In the first, **Grass** is more dominant than **Urban** whereas in the second the opposite applies. These mixtures would therefore both appear in the table under the **Bare Soil** row. Percentage cover is not considered. If coverage is equal between classes, one selection is listed, chosen arbitrarily.

The table can be used to establish, for example, that

- **Sand** only exists as a *pure* (100% coverage) class;
- **Stones**, which is not underlined and is not the dominant class of a mixture, only exists as part of a mixture, as a secondary class at most;
- the classes which mix the most are **Bare Soil**, **Grass** and **Shrubs** followed by **Coniferous** and **Broadleaf** classes;
- **Fruit Trees** is not often dominant in a mixture but appears quite often as a third or fourth component, similarly to **Vines** which, with **Horticulture**, is always mixed with **Bare Soil**;
- the other classes, **Water**, **Marsh**, **Urban**, **Stubble** and **Cereals** only exist as part of two component mixtures or as *pure* classes.

However, although table 4-4 is useful for a rapid overview of the mixing of classes, it does not provide quantitative composition information such as the number of pixels of each class. For simple data sets, such an analysis can be carried out manually but for this very large data set, it was necessary to develop an automatic method for representing the mixture information from the reference data. For this purpose, a so called 'Composition matrix' was developed.

The *Composition* matrix for the 'Portugal fifteen class' data set is shown in figure 4-16. The matrix lists the *pure* class numbers vertically (A). Horizontally (B), it lists the possible positions within a mixture. For example, if there is a maximum of four components mixing in the reference data, then there are four possible positions in which a class can lie. The matrix is filled by counting the number of times a class occurs in each position. For example, in a [60% class 5 + 40% class 14] mixture, the counters for the

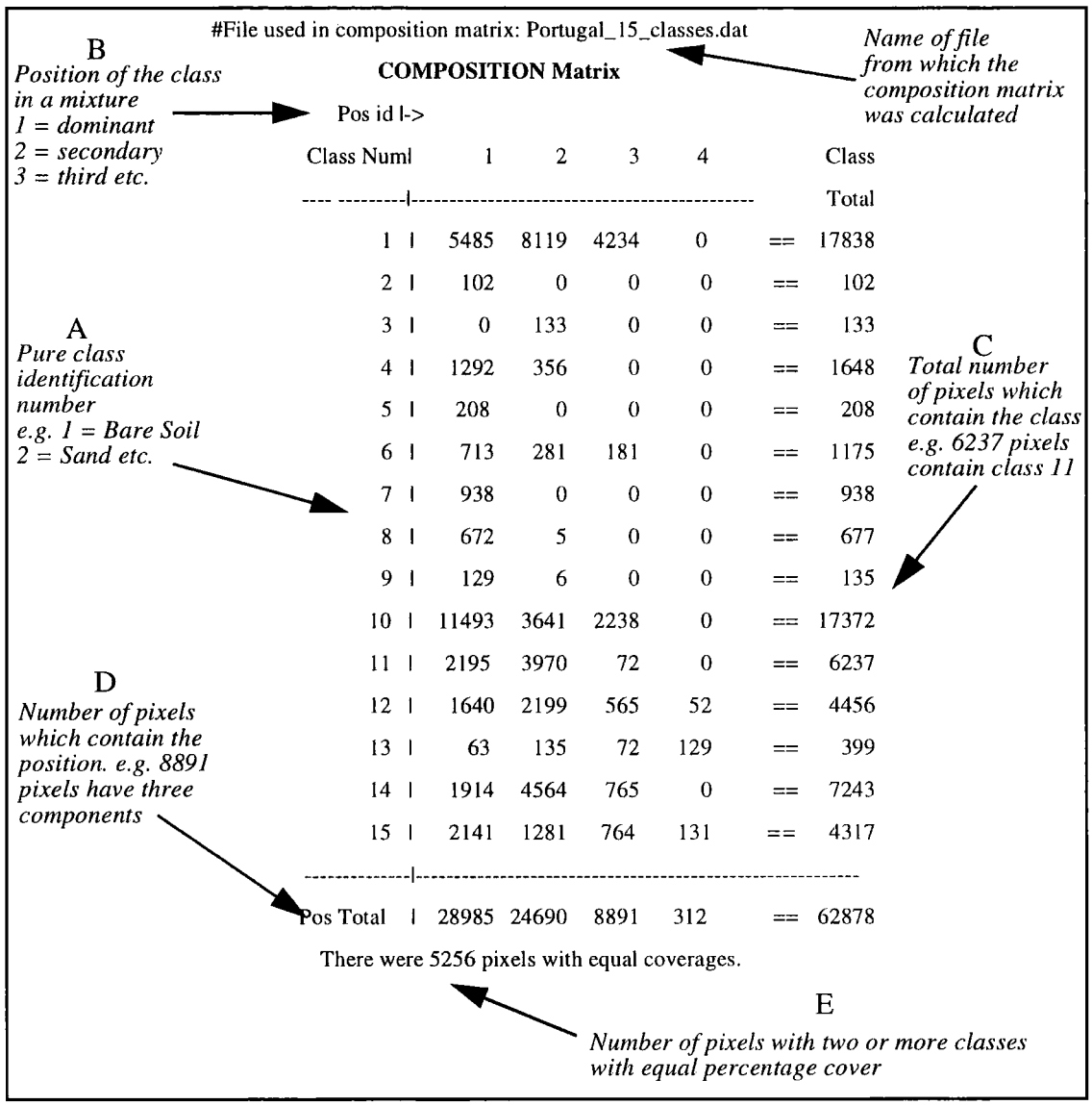


Figure 4-16. Composition matrix for the 'Portugal fifteen class' data set

dominant position of class 5 and the secondary position of class 14 would both be increased by one. If two or more classes cover the same proportion of a pixel, for example, [30% class 1 + 40% class 2 + 30% class 6], then both (or more) class counters for the relevant position are incremented by one; in the example, the counters in position two of classes 1 and 6 would both be increased. The column on the far right entitled 'Class Total' shows how many pixels contain each of the classes, irrespective of the class position in the mixture (C). This provides an idea of the *a-priori* probabilities of each class. However, if two classes are present in the same number of pixels but one class is mainly present as a dominant class whereas the other is mainly present as a secondary class, the *a-priori* probabilities are not the same. The matrix also provides a breakdown of the position of classes within pixels. The last row of the matrix (D), entitled 'Pos. Total' (Pos. = Position) provides, for each column of the number of components, the number of pixels which

contain that number of components. Finally, the last line (E) indicates the number of pixels which had two or more classes with equal percentage cover.

For example, from this composition matrix, it is possible to tell the following.

- **Sand**, **Marsh**, **Stubble**, **Cereals**, and **Horticulture** exist almost exclusively as dominant classes within a mixture.
- **Fruit Trees** mainly exists as a second (34%), third (18%) or fourth (32%) component of a mixture; only 16% of pixels containing the class **Fruit Trees** have it as a dominant class.
- **Deciduous Forest** occurs as a secondary class 66% of the time whereas it occurs as a dominant class only 26% of the time.

In other words, the composition matrix allows the user to calculate the percentage of each class within the data set, the percentage of each class at each position of a mixture and other similar statistics.

4.5 'Portugal seven class' Data Set

The 'Portugal fifteen class' data set which was created for the purpose of this thesis is complex and large which makes its analysis difficult. A smaller, further simplified data set containing seven *pure* classes was created with limited numbers and types of mixtures.

4.5.1 Methodology for creating the data set

The 'Portugal fifteen class' data set was reduced by:

1- only considering those classes which had at least one polygon with 100% coverage,

2- from the above, only considering those classes in two component mixtures,

3- ignoring **Stubble** as a class because of its ambiguous definition,

4- defining the **Vines** class as a homogeneous mixture of soil and vine plants and changing proportions in consequence. For example, [30% **Vines** + 70% **Bare Soil**] becomes 100% **Vines**.

The final data set included **Bare Soil**, **Sand**, **Water**, **Urban**, **Cereals**, **Grass** and **Vines** as *pure* classes. 11,852 pixels were extracted. The individual classes and the mixture types are listed in table 4-5. The composition matrix for this data set is listed in appendix B, section B.6. It was not necessary to have a set of classes which were representative of the original landscape since only accuracy was of interest here.

Id.	Class name	Pure Types	Number of Pixels	Mixture Types	Number of Pixels
1	Bare Soil	100% 1	984	20% 1 80% 6	1,063
2	Sand	100% 2	102	30% 1 70% 6	225
3	Water	100% 3	1,196	40% 1 60%06	248
4	Urban	100% 4	528	70% 1 30% 6	270
5	Cereals	100% 5	372	80% 1 20% 6	714
6	Grass	100% 6	2,827	20% 6 80% 7	1,419
7	Vines	100% 7	1,024	40% 6 60% 7	718
				20% 3 80% 6	162
Total Number of Pixels					11,852

Table 4-5. Composition of the 'Portugal seven class' data set

4.5.2 Analysis of the data

Spectral Analysis

The smaller number of mixtures, compared to the 'Portugal fifteen class' data set, allows the spectra for each mixture type to be displayed. Figure 4-17 shows the spectra for each of the categories listed in table 4-5. The spectra have been arranged so that mixtures are ordered between the *pure* spectra that compose them, in decreasing order of the percentage of the class on the left and increasing order of the percentage of the class on the right. Classes 2, 4 and 5 do not exist within mixtures in this data set. The categories and their spectra presented in figure 4-17 have been divided up for a more detailed analysis below.

Figure 4-18 shows the spectra for each of the classes in their *pure* state. Some of the classes have a large spread of values in each of the bands, particularly **Bare Soil**, **Sand**, and **Cereals**. The **Water** class seems, as before, to be composed of two distinct signatures, one of which is vegetation, which implies poor definition of the polygons at the ground campaign stage. It is possible that some of the **Water** polygons were mapped in areas of rice fields even though no vegetation was recorded on the campaign sheet. The **Grass** class has a number of outliers in the lower part of the spectra which suggest that there may be non-grass pixels within this class. The **Vines** class has less spread in its values than the **Bare Soil**, **Sand** and **Cereals** classes and no outliers.

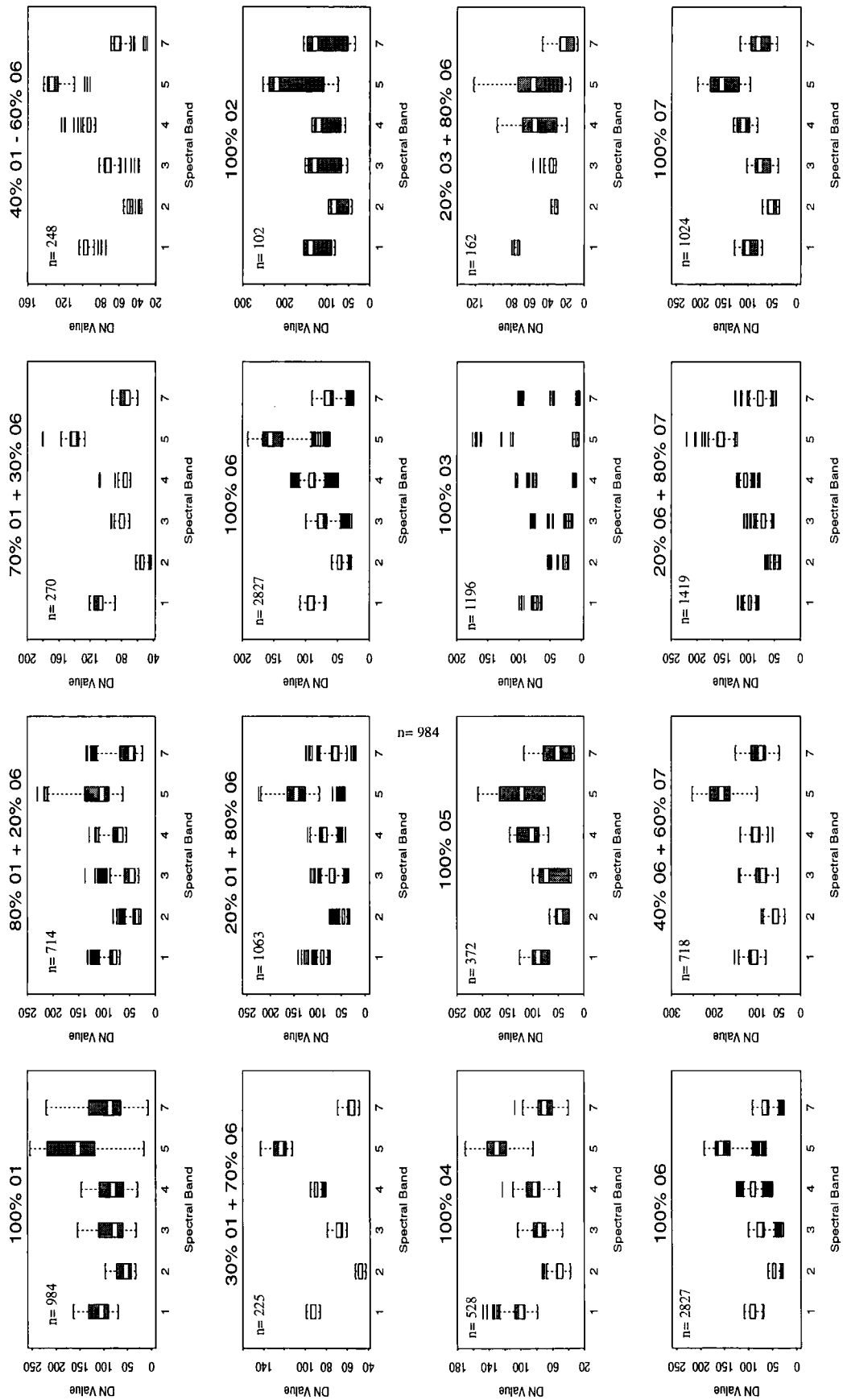


Figure 4-17. Spectra for the 'Portugal seven class' data set

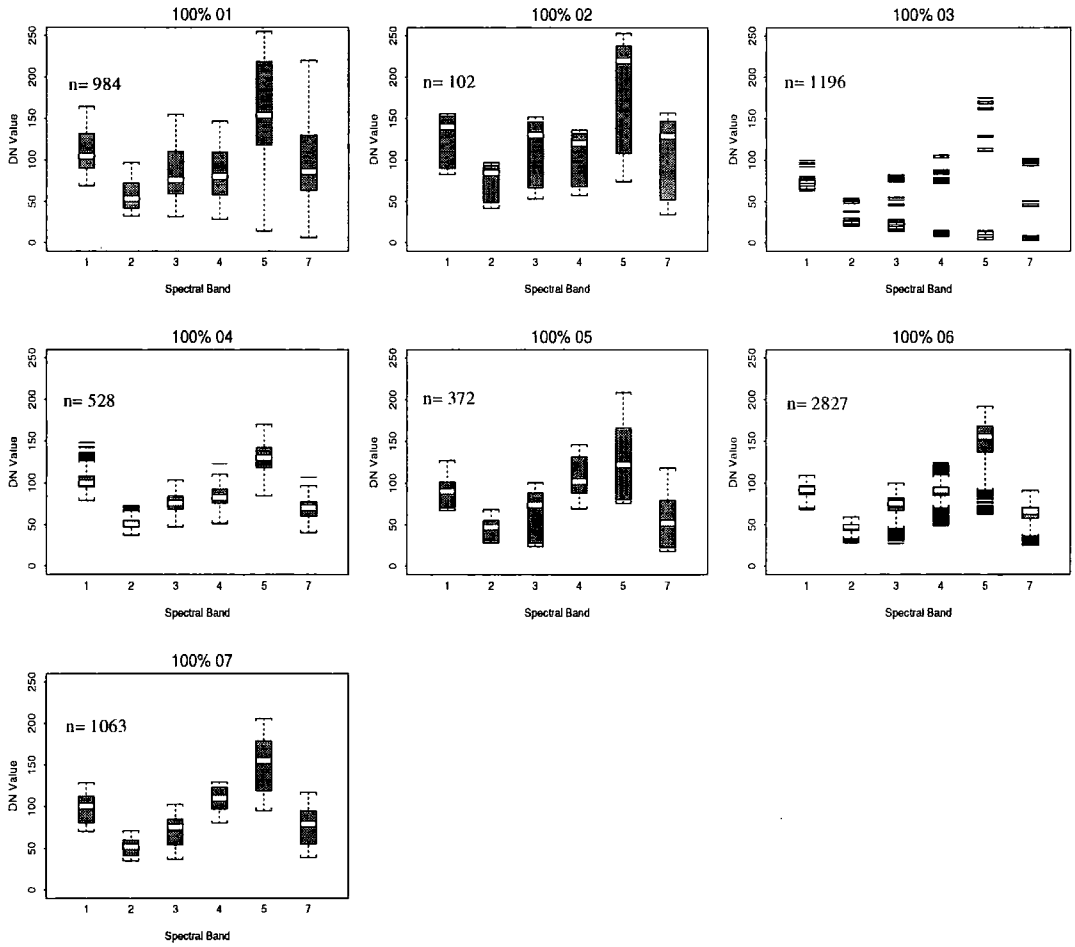


Figure 4-18. Spectra for the 100% pixels

Figure 4-19 shows the spectra for **Water**, **Grass**, and mixtures in between. [20% **Water** + 80% **Grass**] is actually more similar to the *pure* **Water** spectra, excluding the outliers, than it is to the **Grass** class. This indicates that either the percentages indicated in the ground survey are wrong or **Water** and **Grass** mix non-linearly. Linear mixing implies that the classes affect the spectral signatures proportionally to the percentage of the pixel they cover. If the percentages recorded in the field survey are correct, then it

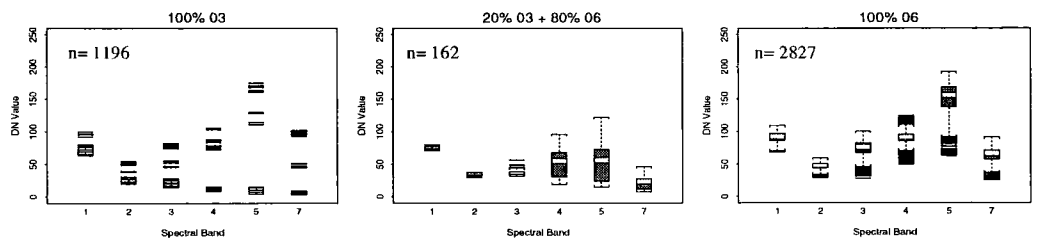


Figure 4-19. Spectra for 100% Water, 100% Grass and mixture in between

seems that the presence of **Water** has a stronger influence on the composite signal from the mixture than does the **Grass** class.

The spectra in figure 4-20 show the mixtures of **Grass** and **Bare Soil** and appear to confirm that mixing between classes is non-linear. If the classes mixed linearly, the expected pattern of the signatures would be a gradual transformation from one class into the other. As the percentage of **Bare Soil** decreased and that of **Grass** increased, the spectra would be expected to look less like **Bare Soil** and more like **Grass.** This is not the case. The mean of the spectra at each band drops significantly with only 20% less **Bare Soil** than a *pure* pixel and the spectra are not that close to the *pure* **Grass** signal either.

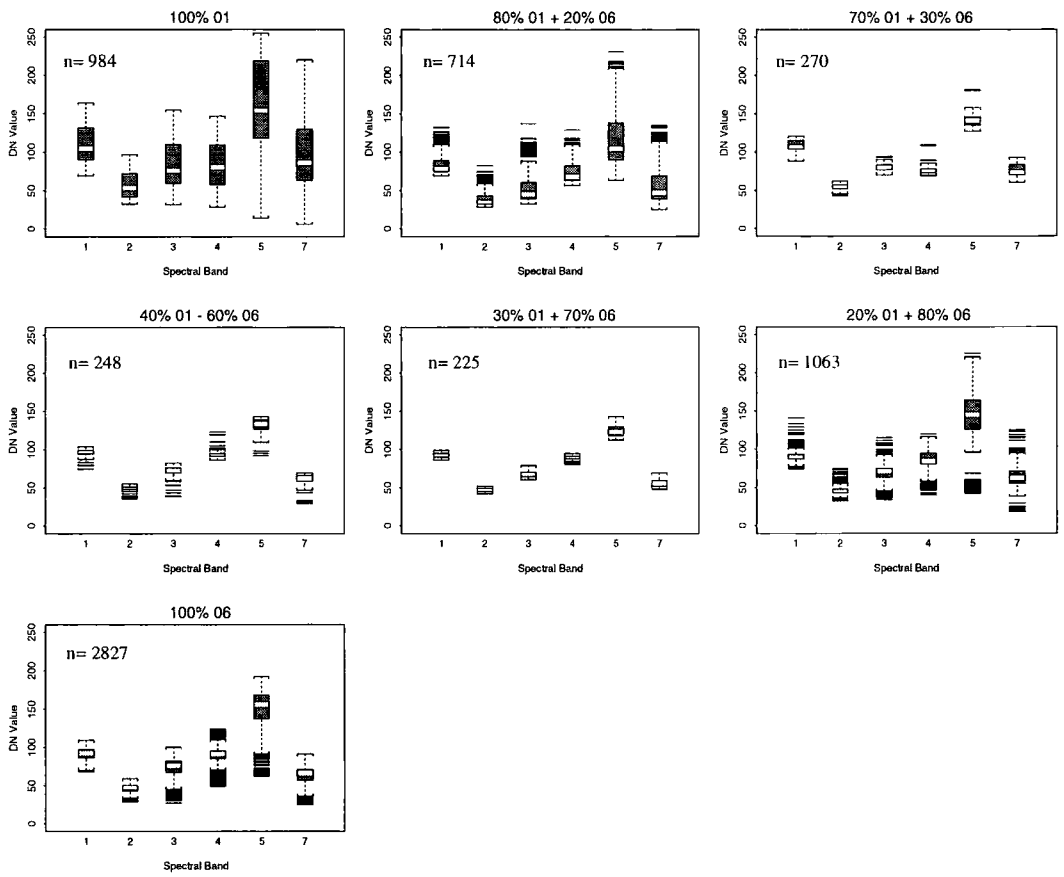


Figure 4-20. Spectra for 100% Bare Soil, 100% Grass and mixtures in between

Finally, figure 4-21 shows the spectra for a mixture of **Vines** and **Grass**. Again, the spectra of the mixtures do not lie between the spectra of the *pure* classes which compose them. For example, the mean of band 5 for the mixtures is higher than the means of band 5 for either *pure* components.

The fact that the spectra do not appear to mix linearly suggests that a linear mixture model may not be appropriate to this problem. It also implies that the network will have to find a mapping between class coverage and spectral signal which is not obvious in these graphs.

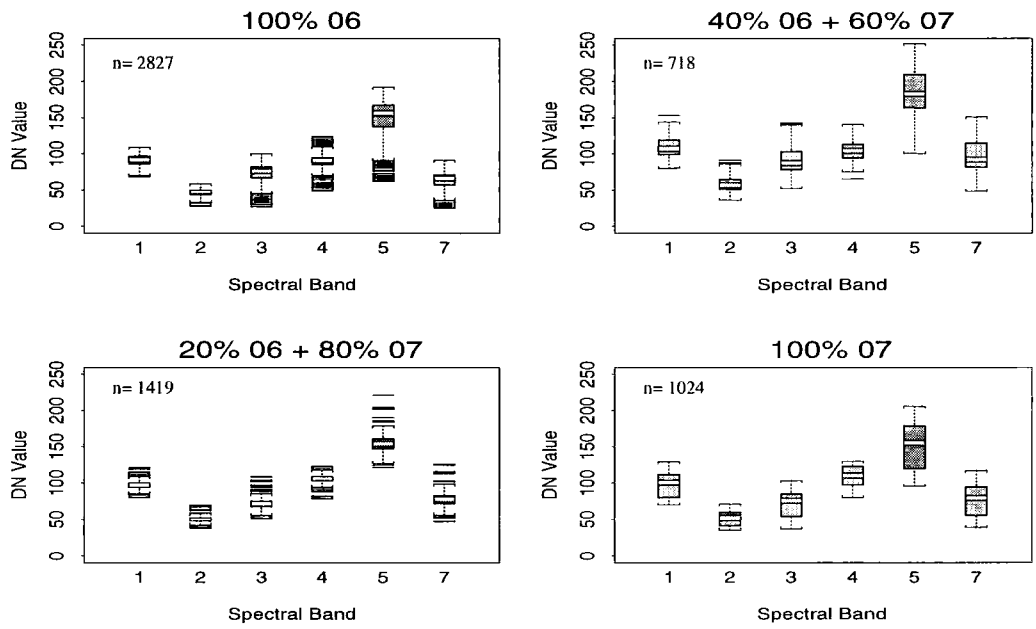


Figure 4-21. Spectra for 100% Grass, 100% Vines and mixtures in between

The description and analysis of the data sets from the Portugal site is now complete. As the description shows, the production of data sets on which to train and test a classifier can be extremely time consuming. The complexity and size of the data sets from the Portugal site have been highlighted. Quantile-quantile plots have shown that data may not be normally distributed and spectral box plots have shown that mixing may not be linear, implying that other classification techniques such as the maximum likelihood classifier and the linear un-mixing model may not be appropriate.

The method for creating the data sets caused some difficulties. For example, the two parcels shown in figure 4-22 and figure 4-23 are both defined as 100% bare soil. Consequently, even though pixels from these parcels will have different spectral signatures, they cannot be differentiated in the data set. Furthermore, the visual interpretation of percentage cover by different field staff produced inconsistencies in the data. For example, figure 4-24 and figure 4-25 show photographs of vineyards and their description. Although the difference between the vine content of figure 4-24 and figure 4-25 is only 2% according to the interpreters, from the photographs, the difference appears to be larger; for the photograph in figure 4-25 (and albeit not from on the ground), the author would have assigned a percentage of 15-20% of vines rather than the 5% assigned.

For these reasons, which later translated into difficulties in the interpretation of results of the experiments, another site was chosen, from which precise and accurate data sets could be created. These are described in the next section.

Figure 4-22. Photograph showing a parcel described as containing [100% bare soil]

Figure 4-23. Photograph showing a different parcel described as containing [100% bare soil]

Figure 4-24. Photograph showing a parcel described as containing [3% vines + 97% bare soil]

Figure 4-25. Photograph showing a parcel described as containing [5% vines + 5% weeds + 90% bare soil]

4.6 Collection of Reference Data for Scotland

A number of studies, mostly carried out in areas of natural forest, have shown that parameters of coniferous forests such as height (Franklin, 1994), mean diameter at breast height (Danson and Curran, 1993) and tree growth (Ahern *et al.*, 1991), and derived variables such as basal area (Brockhaus and Khorram, 1992) and stand density (Horler and Ahern, 1986) are significantly correlated with reflectance signals picked up by the Landsat TM sensor. In fact, changes to these variables, for example from growth of the trees, reflect changes in the relative proportions of trees and background; that is, the extent of canopy closure. Background vegetation and soils generally have a higher reflectance than conifer canopies (Franklin, 1986; Gemmel and Goodenough, 1992) and so the reflectance measured by the sensor decreases as the percentage tree cover increases. This can be used to advantage to study the presence or absence of the understory vegetation (Stenback and Congalton, 1990). It is therefore hypothesised that a study of the mixing between coniferous trees and grass can substitute measurements of the proportions of sub-pixel components with measurements of structural variables of the forest.

Most studies reported in the literature have been carried out in areas of natural forest. Although the structural variables which describe the forest are the primary source of changes in reflectance, other sources that affect the signal of the forest cover include slope, tree spacing, species composition, type of background and tree health. The site chosen for this study is situated in the Dumfries and Galloway region of Scotland and its typical landscape is shown in figure 4-3. It consists of plantations of coniferous forest and has the following advantages over a natural forest area:

- almost unique species composition (mainly Sitka Spruce);
- forest understory (when present) consisting almost exclusively of grass;
- regular and dense planting;
- simple single storied homogeneous canopies with few gaps;
- most trees planted on flat or gentle slopes;
- minimal thinning and, whenever possible, immediate control of disease and poor growth.

In addition, only a small proportion of the image is used and the scene is of good quality with little atmospheric influence. The relationship between structural parameters and canopy reflectance can therefore be expected to be strong and a high degree of confidence in their accuracy, associated with the data sets.

Ancillary data for the area included

- 1: 10,000 colour aerial photographs (1992);
- 1: 10,000 forest stock maps (1980's mainly).

These were made available to the Department of Geography by Forest Enterprise, UK. Stock maps are produced by the Forestry Commission and identify forest compartments in terms of extent, year of planting and species composition. Unless trees are harvested or thinned, the planting date and species composition of parcels provided on stock maps can be taken as correct. Aerial photographs are invaluable for estimating the homogeneity of an area and the degree of closure of the canopy, both of which are often impossible to tell from the ground. The aerial photographs predate the 1995 image and postdate the 1989 image by three years. Although this should be taken into account, particularly for younger trees, three years is a short time in terms of tree growth and should therefore have a minimal effect on interpretation.

From these data, a first reference data set was created by delineating areas on stand maps which appeared to be homogeneous according to the aerial photographs. Areas of closed canopy, medium density and low density trees were chosen. As the density decreases so does the homogeneity. Boundaries of the polygons were drawn at a distance of more than 30m from the stand edges to eliminate the need to buffer the polygons later. These data are typical of conventional *pure* classification data; each polygon is assigned to only one class.

In addition to these data, a forest survey was carried out by the author and staff from the Department of Geography of Durham University, in May 1996. Only Sitka Spruce stands were chosen to limit the effect of species differences on the signal variability of the pixels. The ground survey was carried out as follows:

1- Prior to the field survey, using aerial photographs, homogeneous areas were identified. Although random sampling may theoretically be the best methodology, practical considerations such as accessibility often limit this method and polygons to be surveyed were also considered for ease of access. The polygons were delineated on the stand maps, at a distance of more than 30m from stand boundaries, and digitized.

2- (a) In the field, at each site to be surveyed, plots were located at more than 30m from the stand edge.

(b) Once within the area to be surveyed, a marker was thrown randomly to represent the centre of a plot. A circle of 5.3m radius was marked out within

which measurements were taken. As mentioned in a separate study from the same research group (Puhr, 1997), surveying larger plots was impractical considering the difficult working conditions found in the plantations. However, the homogeneity and simplicity of the forest should allow the plots to be representative of the whole stand.

3- (a) For every tree in the plot that was not dead, the species was noted and Diameter at Breast Height (BH = approximately 1.3 m) measured. Dead trees are not included as they do not contribute significantly to the canopy's spectral response. Each stem of multi-stemmed trees was measured.

(b) Using a clinometer, which measures the angle between the observer and the object being pointed at, measurements were taken for the height of the thickest tree, or another if visibly higher. If the top of the tallest tree was not visible from within the stand, measurements were taken from outside. The distance and the slope between the observer and the tree was recorded.

(a) Slope and aspect of the plot were recorded together with any comments concerning the plot.

4- Where possible, since signals were not received within closed canopies, a position was read using a Global Positioning System - it was later estimated to be accurate to within 10m.

In total, 16 sites were visited; in some two to three plots were measured. Appendix B, section B.7. shows a survey sheet for one of the plots. A sample aerial photograph used in the survey, on which the approximate location of the digitized polygons and polygon reference numbers are indicated, is provided in figure 4-26. Spatial location of plots was very well controlled with the use of these large scale photographs, maps and a Global Positioning System.

4.7 'Scotland Eight class' Data Set

4.7.1 Methodology for creating the data set

A data set was created from the 1989 image and the homogeneous polygons of *pure* classes described in the previous section which will be referred to as the 'Scotland eight class' data set, whose composition is listed in table 4-6. Classes are considered to be *pure*. For example, pixels which belong to the class **Low Density Trees** are covered by a mixture of trees and grass but the scale of mixing is considered to be less than the size of the pixel, so that the percentage cover of the polygon by each class, trees and grass, is

Figure 4-26. Sample aerial photograph showing the approximate position of some digitized polygons for location in the field.

Id.	Class Description	Number of Pixels
1	Closed Canopy Sitka Spruce	189
2	Closed Canopy Lodgepole Pine	209
3	Water	267
4	Background (Grass, Rocks)	637
5	Deciduous	418
6	Low Density Trees (mainly Sitka Spruce)	376
7	Medium Density Trees (mainly Sitka Spruce)	208
8	Closed Canopy Trees (Mixed)	425
Total number of pixels		2,729

Table 4-6. Composition of the 'Scotland eight class' data set

representative of that of each individual pixel. 2,729 pixels were extracted from the 1989 image to form this data set.

4.7.2 Analysis of the data

The spectra for each of the eight classes were plotted and are shown in figure 4-27.

- The **Water** class has a different signature from all the others and can therefore be expected to be easily separated.
- The **Closed Canopy Sitka Spruce** and **Closed Canopy Lodgepole Pine** classes are differentiated spectrally by a higher median value for **Closed Canopy Sitka Spruce** in bands 4 and 5 particularly. Both classes have some outliers.
- **Low Density Trees** and **Medium Density Trees,** which are mainly Sitka Spruce, are differentiated from their closed canopy equivalent by higher values in all the bands and particularly bands 4, 5 and 6. The median for **Medium Density Trees** are slightly lower than for **Low Density Trees**. The latter class has no outliers, the former class has a few.
- The **Closed Mixed (SS-LP)** class has values which are between the **Closed Sitka Spruce** and **Closed Lodgepole Pine** spectra, with few outliers.
- The **Background** class has higher spectral values in all bands. It also has a larger spread of values and is less well defined.
- Finally, the **Deciduous** forest class has the highest median in band 4 of all the

classes and a median in class 5 which is higher than for the closed canopy classes and the mixed canopy classes but lower than the **Background** class. From these spectra, a relatively good accuracy of classification can be expected as classes seem well defined and separable.

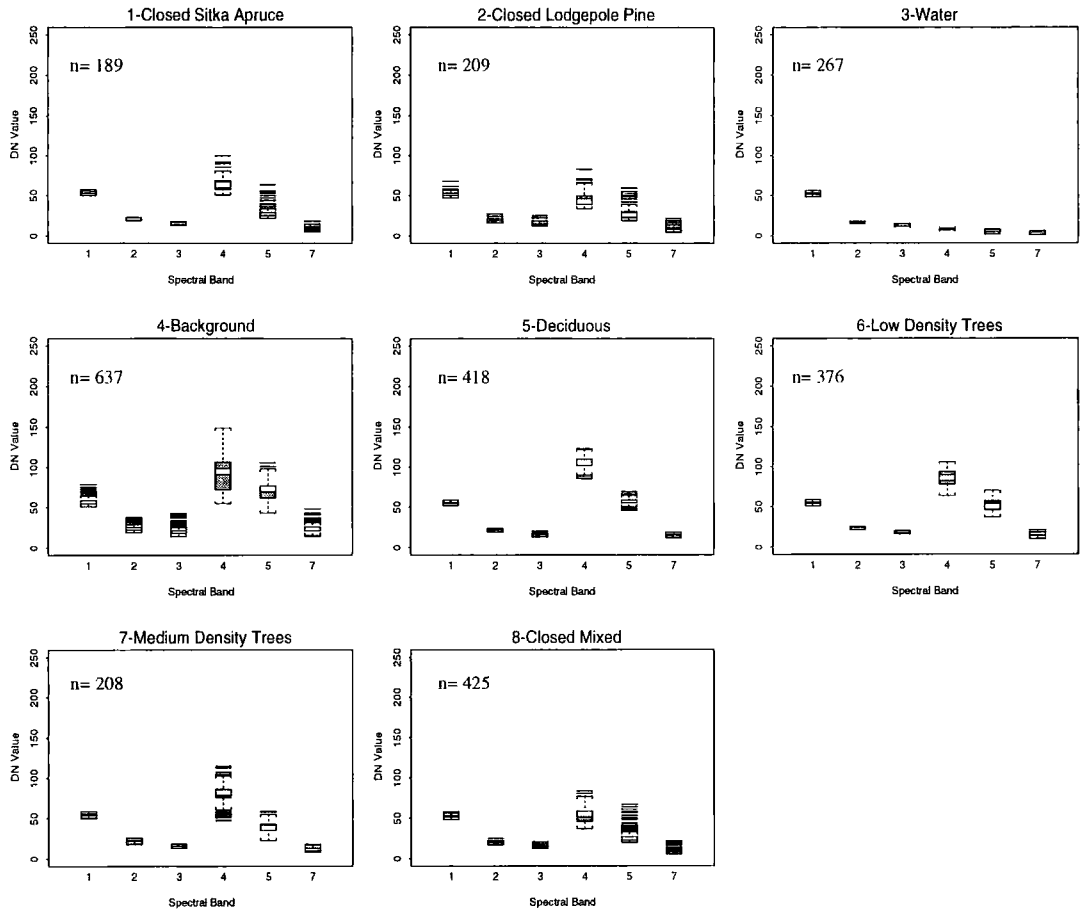


Figure 4-27. Spectra for the 'Scotland eight class' data set

4.8 'Scotland Bio All' and 'Scotland Bio Box' Data Sets

4.8.1 Methodology for creating the data sets

From the measurements taken in the field, as described in section 4.6, Basal Area, Height and Stem Density values were calculated for each digitised polygon. A summary of the values is provided in table 4-7; when there were several plots, values for each plot were averaged.

1- Basal Area is the total area within the plot which is covered by tree trunks. It is calculated using the diameter at breast height (DBH) for each tree. Basal Area for each stem of multi-stemmed trees are included.

2- Height is calculated from angle measurements using trigonometry. If the ground is not level, the effect of the slope should be taken into account.

Polygon Visit Id.	Height in m	Basal Area in m^2	Stem Density in trees / ha	Year of Planting
1	12.75	39.51	3,060	1960
2	15.56	71.94	2,946	1963
3	18.04	57.28	2,606	1962
4	7.00	36.57	1,700	1976
5	5.31	6.34	1,303	1979
6	6.17	21.00	1,700	1979
7	5.42	17.57	1,813	1979
8	17.76	62.35	3,060	1955
9	13.60	56.45	1,926	1971
10	18.64	56.76	3,683	1956
11	5.27	15.21	1,586	1981
12	5.68	22.00	1,926	1981
13	4.49	9.73	1,586	1981
14	5.16	9.99	1,700	1981
15	7.34	26.46	1,813	1975
16	17.56	56.17	2,776	1948

Table 4-7. Summary of the variables calculated from field measurements for the Scotland site

3- Stem Density values are calculated by counting the number of trees in the plot; multi-stemmed trees are counted as one. Two data sets were created from the 1995 image: 'Scotland bio all' and 'Scotland bio box'.

(a) For the first data set, 'Scotland bio all', all the pixels in every digitized polygon were extracted along with their corresponding structural measurements. Then *pure* Grass pixels were added. The size of this data set was 936 pixels. Upon analysis of the data, variability was found to be quite large and it was decided to create a second data set.

(b) The second data set, 'Scotland bio box', was created by extracting only the channel values for pixels within a 5x5 pixel box. The box was preferably centered at the location of the plot from which measurements were taken, but if it was considered that there was an area more homogeneous than that around the plot location, still within the forest stand, the box was located in the most

homogeneous area. *Pure* Grass pixels were added. The size of this data set was 515 pixels. The *pure* grass pixels that were added to both data sets are the same.

4.8.2 Analysis of the data

The variables were plotted against each other as shown in figure 4-28. Despite the relatively few points mapped, the graphs in figure 4-28 show that there appears to be an almost linear relationship between the variables which becomes less evident at the extreme values, that is for older trees. As discussed in section 4.6, it is thought that structural parameters of the forest such as Height and Basal Area can be related to the spectral signature of the pixel because the sensor measures the background reflectance. Young trees have a large amount of reflectance from background vegetation whereas older trees have a small amount of background reflectance. When canopy closure occurs, that is, when background can no longer be viewed by the sensor, the relationship breaks down. Peterson *et al.* (1986) also found saturation effects in the response from the clustering of older trees.

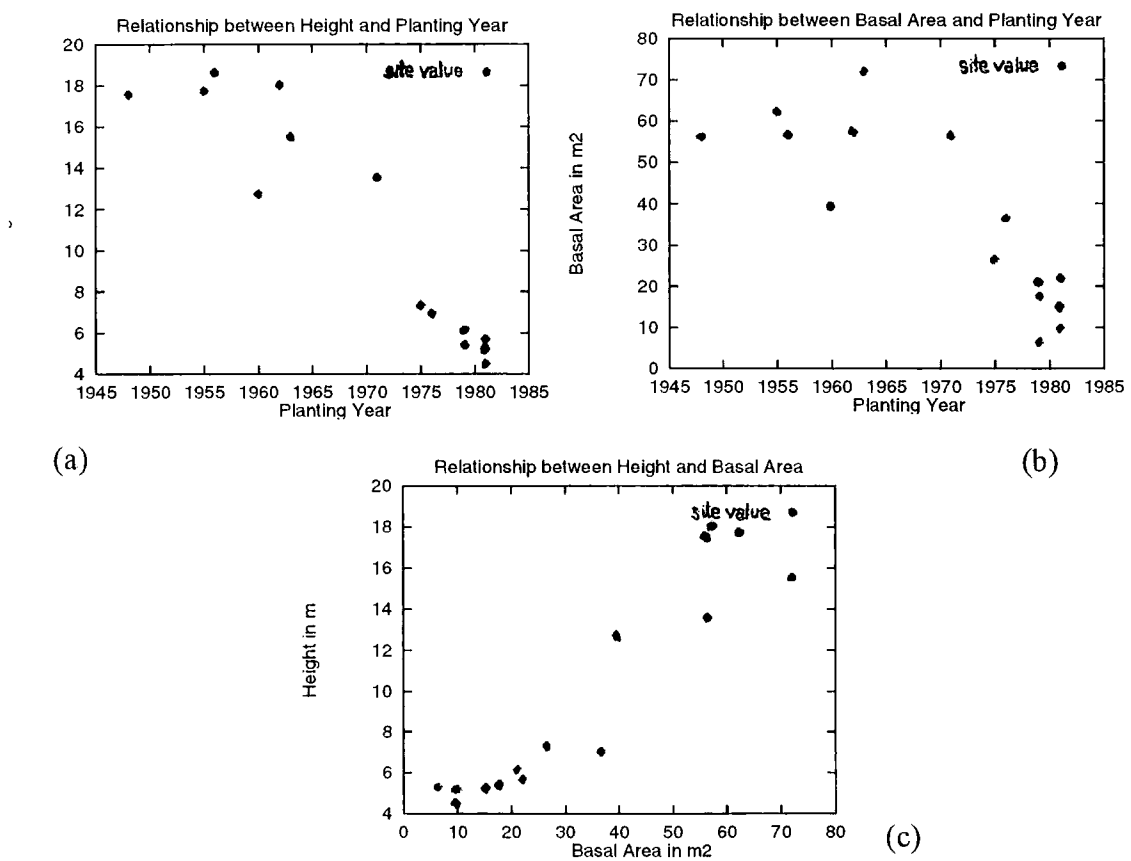


Figure 4-28. Graphs of Basal Area against Year of Planting (a), Height against Year of Planting (b) and Height against Basal Area (c)

Pearson correlation coefficients, which provide a measure of the linear correlation between variables, were calculated for Height, Basal Area, Stem Density and planting Age and are provided in table 4-8. From the table, Stem Density seems to have the weakest relationship with Basal Area, and Basal Area has the weakest relationship with Age. Height has the strongest relationship with Basal Area and Age. Puhr (1997), from the same forest, found strong positive correlations between Basal Area and tree Height, tree Height and stand Age, and Basal Area and stand Age; and weaker correlations between Stem Density and the other variables. Since Age is not a structural parameter, it was not studied further in this thesis.

Variable	Basal Area	Height	Stem Density	Age
Basal Area	-	0.93	0.81	0.84
Height	-	-	0.89	0.94
Stem Density	-	-	-	0.89
Age	-	-	-	-

Table 4-8. Pearson correlation values for the relationship between the structural parameters

The mean DN values of each channel were plotted for increasingly large square areas from a point to study the variability of the reflectance of pixels within each polygon. Changes to the mean signal are an indication of the heterogeneity of the area being studied. In the case of the first data set, 'Scotland bio all', the mean reflectance values were extracted for increasing kernel sizes (1x1, 3x3, 5x5, 7x7...) in proportion to the size of the polygon. Thus, if a polygon was small, it may only include a 3x3 kernel; if it was large it may include a kernel up to 7x7 or more.

The graphs showed some heterogeneity in the data. For this reason, the second data set, 'Scotland bio box', was created. Only kernels of 3x3 and 5x5 were studied since all pixels in the data set were extracted from 5x5 boxes. The graphs, showing mean DN values for each channel, for increasing kernel sizes, are provided in appendix B, section B.8, but a sample are shown in figure 4-29.

Figure 4-29.a and figure 4-29.b show two graphs from the 'Scotland bio all' data set and figure 4-29.c and figure 4-29.d show the graphs for the 'Scotland bio box' data set for the same sites. From figure 4-29.a and figure 4-29.b, it is apparent that in plot 77, the 3x3 and 5x5 means are similar whereas the 7x7 means have decreased. On the other hand, the 3x3 means in plot 78 are slightly higher than the 5x5 means. The graphs in figure 4-29.c and figure 4-29.d, on the other hand, show that the variability has been reduced using the

chosen methodology, since the 3x3 means and the 5x5 means are almost the same, implying better class signature definition. The graphs shown are representative of the plots provided in appendix B, section B.8.

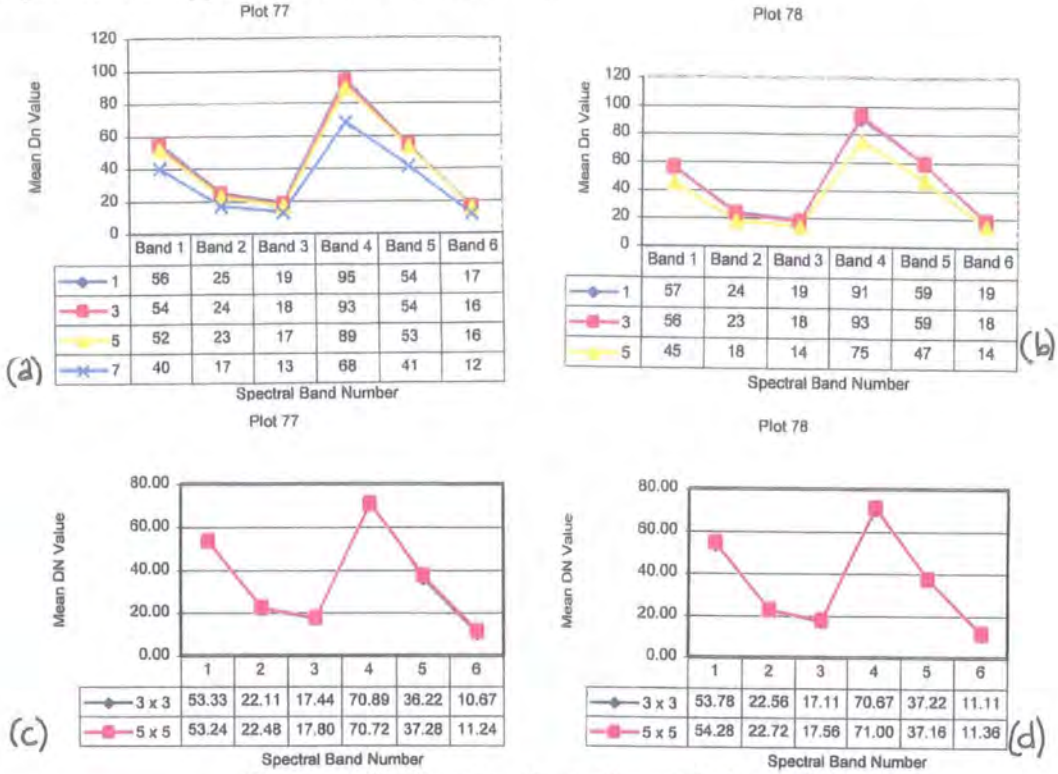


Figure 4-29. Sample of graphs used to evaluate the heterogeneity of polygons from the 'Scotland bio all' and 'Scotland bio box' data sets

Finally, band DN values were plotted against Height and Basal Area variables. Graphs were plotted for the 'Scotland bio all' data set, shown in figure 4-30.a and figure 4-30.b, and for the 'Scotland bio box' data set, shown in figure 4-30.c and figure 4-30.d. The graphs show that reflectance values generally tend to decrease as Height and Basal Area increase. This effect can be attributed to the contribution of a grass signal in the younger plots which increases the reflectance values of the pixels from those plots. The graphs for the 'Scotland bio box' data set show a decrease in the spectral variability for each plot, as expected.

Correlation values between the channel values of the 'Scotland bio box' data set and Height, Basal Area and Stem Density variables are presented in table 4-9. They show that the weakest relationship is with band 1, followed by bands 4 and 7. The highest correlations are found with bands 2, 3 and 5. These results are slightly different from those of Puhr (1997) who found that Height was most correlated with bands 2, 3 and 5 as here, but he found Basal Area to be most correlated with bands 3, 5 and 7. Since the correlation values for Stem Density were relatively low, and since Stem Density in these plantations is mainly a function of planting practice, Stem Density was not studied further in this

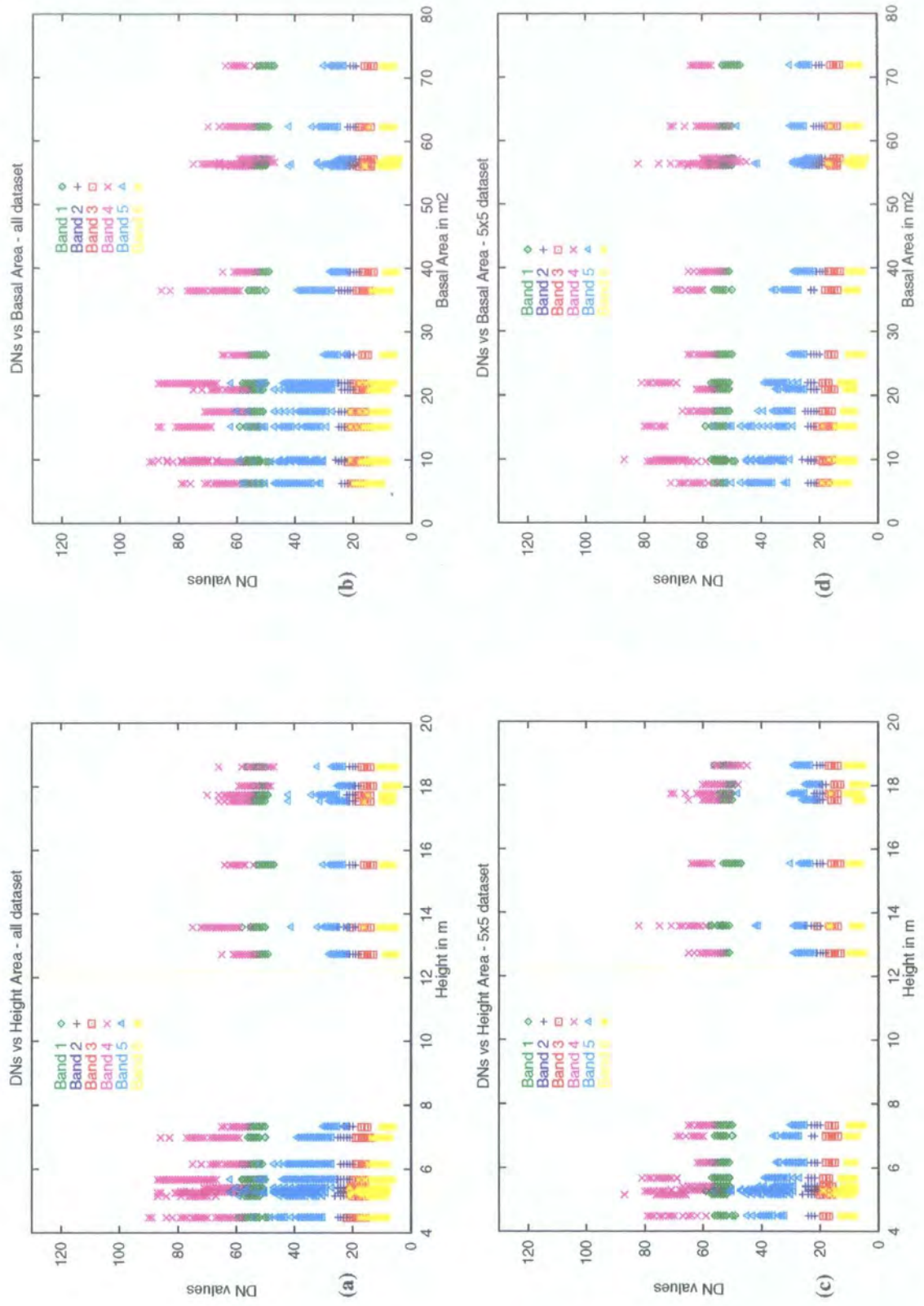


Figure 4-30. DN values vs. Height and Basal Area for the 'Scotland bio all' data set (a & b) and the 'Scotland bio box' data set (c & d)

Variable	Height	Basal Area	Stem Density
Band 1	-0.51	-0.52	-0.48
Band 2	-0.80	-0.74	-0.74
Band 3	-0.72	-0.72	-0.67
Band 4	-0.64	-0.54	-0.64
Band 5	-0.71	-0.73	-0.65
Band 7	-0.59	-0.62	-0.54

Table 4-9. Pearson correlation values for the relationship between the Landsat TM channel values, and Height and Basal Area calculations for the 'Scotland bio box' data set

thesis. The analysis of the forest reflectance suggests that a neural network should be able to map input DN values to the appropriate Basal Area and Height classes with a relatively high level of accuracy.

4.9 Summary of Chapter IV

This chapter has provided a detailed description of the methodology for creating each data set used in this thesis: 'Portugal sixteen class', 'Portugal fifteen class', 'Portugal seven class', 'Scotland eight class', 'Scotland bio all' and 'Scotland bio box'. The data sets created from reference data from the Portugal site were described in sections 4.2 to 4.5. The data sets created from reference data from the site in Scotland were described sections 4.6 to 4.8.

The study of the 'Portugal sixteen class' data set, a *pure* data set, shows that classes multi-spectral data for most classes are not normally distributed which consequently implies that a technique such as maximum likelihood classification may not be appropriate. Considering that *pure* classes are likely to be more normally distributed that mixed classes, this conclusion extends to the mixed data sets. Furthermore, the study of the 'Portugal seven class' data set showed that classes do not appear to mix linearly. This implies that a technique such as the linear un-mixing model, already discarded because of the number of classes under investigation, may not be appropriate. The analysis of the data illustrated the complexity of the data collected at the site in Portugal and suggested that it may cause inaccuracies and difficulties at the classification stage. This conclusion, later verified in the experiments which are described in subsequent chapters, suggested the need for accurate ground data for a second experimental site.

The second site was situated in the Dumfries and Galloway region of Scotland.

Whereas the Portugal data was composed of many classes and mixtures, the only mixing studied here was that between Sitka Spruce coniferous trees of different ages and Grass. Because of the characteristics of the site and the use of precise measurements of structural variables as substitutes for proportions of each class, a high level of confidence in the accuracy of the data could be associated with each data set created from the reference data collected.

The next chapters will describe the experiments that were carried out using the data sets discussed in this chapter.

Chapter V

SENSITIVITY ANALYSIS OF THE NETWORK

The overall aim of the thesis is to test the hypothesis that the network can identify sub-pixel information. However, before analysing the output from the neural network, it is necessary to account for factors not related to sub-pixel information that may influence the outcome of experiments. Running the network requires the user to set (1) the number of iterations to which the network is trained and (2) the architecture of the network. It is necessary to establish whether the choices that are made influence the performance of the network. In addition, the initial values of the weights determine the configuration from which training starts, therefore training may be affected by (3) different weight initialisations. The user must also set (4) the learning rate. Finally, observations may be affected by (5) the manner in which the data set is divided into training and testing data sets. The effect of these parameters (they will be referred to as such even though some of them are not strictly speaking parameters) has to be quantified before *soft* classification of pixels by the neural network can be evaluated.

When neural networks are used in the remote sensing literature, authors very rarely present thorough investigations into the sensitivity of these networks; rather the emphasis is usually placed on the performance of the network for a given application. Sensitivity experiments are extremely time consuming both to run and to analyse. Nevertheless, an initial study of the performance of the network over a range of parameter values is essential as it provides the framework within which further experiments can take place.

Exact parameter choices cannot be compared because different networks and data sets require different settings. Nevertheless, it is interesting to contrast general trends in the experiments reported in this chapter and those reported in the literature. The results from this study are compared with two articles of particular reference to sensitivity experiments for networks trained with remotely sensed data: Skidmore *et al.* (1997) and Paola and Schowengerdt (1997).

This chapter is divided into several sections. First, the general design of the experiments is discussed. Then, the methodology and results for each individual

parameter are reported; their effect on the performance of the network is evaluated using graphs which show the overall accuracy of classification of the testing set over a range of values of the parameter. Each parameter is tested by keeping all others constant. The aim of the experiments is to show that observations in subsequent experiments will not be dependent on particular choices of parameters. Results are compared with those reported in the articles by Skidmore *et al.* (1997) and Paola and Schowengerdt (1997).

5.1 General Design of the Experiments

Ideally, for each neural network classification problem, a large number of training and testing runs of the network would be carried out so that the effect of any one parameter (number of iterations, architecture of the network and so on) could either be discarded as insignificant or fully quantified. Parameters could be comprehensively tested so that every possible network configuration could be studied. However, this is not feasible as there are too many possible values for the parameters, too many combinations of parameters and the sometimes lengthy training time of the neural network restricts the number of training runs which can be executed for each experiment. A representative sample of experiments were therefore carried out to study the effect of changes to the following parameters on the classification results

- number of iterations,
- architecture of the neural network,
- initial weight values,
- learning rate,
- composition of the data sets.

For each study, all parameters other than the one under investigation were kept constant. Initial values for the parameters were based on preliminary experiments not reported here and advice from experienced users of the network (Kanellopoulos, personal communication).

Sensitivity experiments could be carried out on simulated data which have the advantage of perfectly known characteristics. However, it is difficult to model data so that they resemble real data enough for results from simulated data to be extrapolated to the real data set. Furthermore, the behaviour of the network specifically trained with the data sets used in the thesis is of interest here, not simply the performance of the network. Therefore, the networks used the 'Portugal sixteen class' and 'Scotland eight class' data sets described in chapter IV, section 4.3 and section 4.7 respectively.

In subsequent experiments, described in chapters VI and VII, other data sets and target types were used. Even though these will produce variations in the behaviour of the network, the variability is expected to be minimal. Although different data sets and target representations may affect the classification accuracy of the network and the details of parameters, the pattern of parameter effects should remain similar. This is confirmed by the fact that similar experiments to the ones reported here were carried out on different data sets by Kanellopoulos (personal communication) and general trends were similar.

The data sets were randomised and then divided into a ratio of number of training pixels to number of testing pixels of 2:1. The composition of each data set is provided in table 5-1 and table 5-2 (Id. = class identification number; TR = number of pixels in the Training file; TS = number of pixels in the Testing file; Total = total number of pixels in the data set).

Id.	Class Description	TR	TS	Total
1	Closed Canopy Sitka Spruce	126	63	189
2	Closed Canopy Lodgepole Pine	139	70	209
3	Water	178	89	267
4	Background (Grass, Rocks)	425	212	637
5	Deciduous	279	139	418
6	Low Density Trees (mainly Sitka Spruce)	251	125	376
7	Medium Density Trees (mainly Sitka Spruce)	139	69	208
8	Closed Canopy Trees (Mixed)	283	142	425
Total number of pixels		1,820	909	2,729

Table 5-1. Composition of the training and testing files for the 'Scotland eight class' data set

Id.	Class Name	TR	TS	Total	Id.	Class Name	TR	TS	Total
1	Tiled / Concrete	221	93	314	**9**	Maize	224	58	282
2	Sand	73	99	172	**10**	Aquatic Plants	373	128	501
3	Bare Soil	365	503	868	**11**	Vineyards	623	423	1,046
4	Sea Water	233	135	368	**12**	Weeds	626	515	1,141
5	Fresh Water	189	88	277	**13**	Garrigue	1,184	281	1,465
6	Estuary / Lagoon	470	251	721	**14**	Grasslands	2,220	1,111	3,331
7	Wheat	267	156	423	**15**	Deciduous	458	370	828
8	Barley	190	121	311	**16**	Coniferous	331	126	457
Total number of pixels							8,047	4,458	12,505

Table 5-2. Composition of the training and testing files for the 'Portugal sixteen class' data set

Both sets of data assume that pixels are covered 100% by one class only. The targets for each pixel are vectors of length ***n*** where ***n*** is the number of *pure* classes. For each pixel, the class to which it belongs is identified by +1 in the target; all other classes are set to -1 (see chapter III, section 3.3.4).The architecture of the network for the Portugal site was set at 6-28-16 and the learning rate was fixed at 0.1 as recommended by Wilkinson *et al.* (1995) who used the same network implementation on the same data sets. For Scotland, the initial architecture was set to 6-15-8 and the learning rate was also set to 0.1

Weights were usually initialised using a constant as a seed value. This is the method used in the original implementation of the network. The routine which calculates random initial weight values uses a constant as a seed. This means that the weight values are random but for the same configuration of the network they remain the same set of random values; this is useful for re-running networks for checking purposes. This routine was modified to allow the option of choosing the computer clock's current time as a seed value. In this way, the initial weight values are different every time the network is run even if the same configuration is kept.

The end of training occurs either because the number of iterations set by the user has been reached or because the system error has fallen below the error threshold set by the user (see chapter III, section 3.1.1). In order that the controlling variable for termination of training would not be the error threshold, this was set to a low value of 0.0001 for all the experiments.

Performance of the network can be measured using the root mean square error of the system during training or the accuracy of classification of the test set, given as the overall percentage of correctly classified pixels. During training, the temporary error on the learning set is logged at every three iterations. In addition, every fifteen iterations, the temporary accuracy of classification of the testing set can be calculated. The temporary error or accuracy numbers are the values that would be obtained if training was stopped at that point. The final error and accuracy values are reached on the last iteration. The temporary accuracy of classification of the learning set is also calculated at every fifteen iterations, but, as explained in chapter III, section 3.3.6, the accuracy of classification of the learning set will be biased since the network has already seen the patterns. For this reason, the learning accuracy was not used to judge the performance of the network.

5.2 Number of Iterations

It was seen in chapter III, section 3.3.6, that the performance of the network is partly a function of the number of iterations with which it is trained. Independently of how many

iterations it is trained with, an inadequate network will not perform well. However, even if a network of adequate size is used, this is not a guarantee of good performance. If the network is trained for too many iterations, it may learn the training data very well but be unable to generalise and classify unseen data; this is termed over-fitting. On the other hand the network may not be able to learn a mapping between inputs and outputs if is trained for too few iterations. The aim of this experiment is to determine how particular choices of iterations affect classification accuracy.

5.2.1 Methodology

In a graph of error against iteration number, the error curve, formed by plotting the temporary error at every three iterations, is generally expected to decrease as iterations increase from 1 to $+\infty$. The curve formed by plotting the overall testing accuracy at every fifteen iterations would be expected to rise. In other words, the network should learn more about the mapping between inputs and outputs with every iteration. However, the rate of learning is rapid at first and then decreases as the network reaches a point at which it has learned all it can. The testing accuracy may then decrease as the network over-learns its training data. The point at which over-fitting occurs cannot be deduced *a-priori*. The point at which the overall testing accuracy no longer rises can be taken as the minimum number of iterations needed to train the network.

Two neural networks, one for the Portugal data and one for the Scotland data, were trained to 3000 iterations. This number was considered adequate to provide an overview of the behaviour of this particular network implementation, but other networks may require much longer training (Paola and Schowengerdt, 1994). The parameter settings for the network are listed in table 5-3, the parameter under investigation is italicised.

Parameters	Portugal	Scotland
Number of iterations	*3000*	*3000*
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 28 16	6 15 8
Type of weight initialisation	Constant seed value	Constant seed value
Learning rate	0.1	0.1
Randomisation	Once	Once
Division ratio	TR : TS = 2 : 1 = 8,047 : 4,458	TR : TS = 2 : 1 = 1,820 : 909

Table 5-3. Parameter settings for the experiment testing the effect of the number of iterations on the overall testing set classification accuracy

5.2.2 Results and analysis

Figure 5-1 shows the curve of the root mean square (rms) error of the system plotted at every three iterations for 0 to 3000 iterations for both the Portugal and the Scotland data sets. The graph on the left has been displayed with a range of [0,8] for the y-axis. The graph on the right shows the same results displayed on a y-axis range of [0.6, 1.2]. This illustrates the differences in interpretation these ranges can cause.

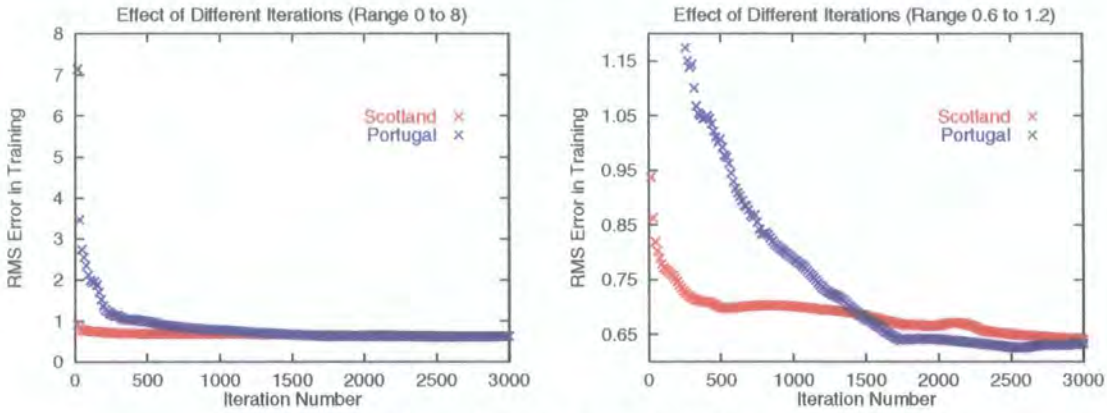


Figure 5-1. Effect of the number of iterations on the rms error; (left) y-axis range [0,8]; (right) y-axis range [0.6-1.2]

Here, from the graph on the left, it would appear that the error on the training set for both the Portugal and Scotland data sets stops diminishing after about 300 iterations. However, on the graph on the right the steepness reduces for the Scotland data set after about 500 iterations and for the Portugal data set after about 1700. For both data sets, the final error approaches 0.65. The error for Portugal starts at a higher value and remains larger than that for Scotland until approximately 1500 iterations when it then reduces to a lower level than Scotland's. The higher initial value may reflect the greater complexity of the Portugal data set: more classes and more pixels. The network needs to find a configuration that represents all fifteen classes whereas for the Scotland data, it only needs to find a configuration to represent eight classes.

When the steepness of the plot of error against number of iterations diminishes, the network is said to stabilize. The fact that the network does not, and apparently will not, converge to an error of zero does not imply that the it has not learned. The final error is a function of the complexity of the data set and of the characteristics of the network.

Figure 5-2 shows the curve of overall percentage accuracy of classification of the testing data set for each data set. The testing accuracy for both data sets seems to stabilize after about 500 iterations but continues to oscillate within about 3% points. The term '% point' is used to differentiate a scalar value from a percentage value: for example, 80% + 5% points would be 85%; on the other hand 80% + 5% would be 84%, the 5% meaning

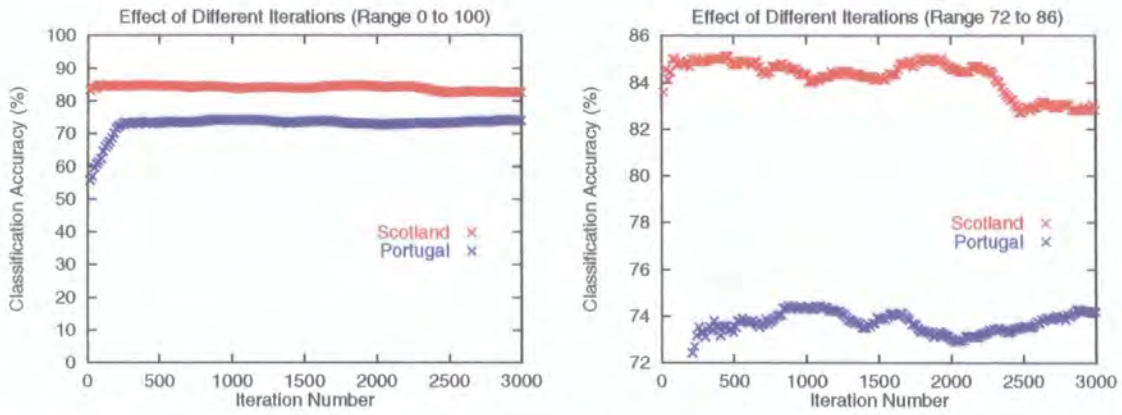


Figure 5-2. Effect of the number of iterations on the accuracy of classification of the test set; (left) y-axis range [0,100]; (right) y-axis range [72,86]

5% of 80%. The testing accuracy for the Scotland data seems, on average, to decrease beyond about 2000 iterations. This could be a sign of overfitting of the data by the network since, as discussed in the previous section, the ability of the network to classify the testing set may decrease if the learning set has been learned too well. The accuracy of the Portugal data of around 73% is consistently lower than that for Scotland of around 84%. This suggests that the problem presented by the Portugal data is more complex than that presented by the Scotland data. However, the difference in accuracies should be carefully evaluated since 73% of 12,000 pixels is actually more pixels than 84% of 2,000.

5.2.3 Summary

The root mean square error on the training set was expected to decrease with increasing numbers of iterations whereas the testing accuracy curve was expected to rise. The results show that the error decreases rapidly and then stabilizes without reaching zero. The testing accuracy curve shows a rapid rise and then relative stability. It may be that the error reaches zero and the testing accuracy reaches 100% with more training but this is unlikely and cannot be seen from these graphs. In any case, the importance of the results is that they show that it is often unnecessary to train for a great many iterations; the accuracy of classification of the testing set after fifteen iterations is already about 55% for Portugal and 80% for Scotland.

The experiments reported in Paola and Schowengerdt (1997) and Skidmore *et al.* (1997) show similar results. Paola and Schowengerdt (1997), who used a network to classify an urban area of Tucson (US), ran their network training for 50,000 iterations. However, after about 8,000 iterations, the accuracy stabilised. Skidmore *et al.* (1997) found no obvious relationship between testing accuracy and the number of iterations.

The error curve in figure 5-1 does not necessarily mirror the testing accuracy curve of figure 5-2 or vice-versa. This is because the error is calculated on the training data whereas the accuracy is measured on the testing data. For this reason, testing accuracy is from now on taken as the only measure of the performance of the network. Considering that the lengthy training time is one of the disadvantages of the neural network approach, and that excessive training can introduce overtraining and loss of generalisation power, it is recommended to keep the number of iterations small. For subsequent experiments, the number of iterations was set at 600.

5.3 Architecture

The architecture of the network, and more particularly its number of hidden nodes, will affect its performance (chapter III, section 3.3.4). Too few nodes means that the network cannot learn the mapping between inputs and outputs. If there are too many nodes, although the network may learn the training patterns well, it will no longer have the ability to generalise for unseen patterns.

The aim of this experiment is to examine the behaviour of the network with different numbers of nodes. It is expected that for a given problem, networks with a large numbers of nodes in the hidden layer(s) would show increasing accuracies of classification until a point beyond which over-training had occurred. After this point, accuracy would be expected to decrease and stabilise.

5.3.1 Methodology

The number of input and output nodes of a network are pre-defined by the nature of the problem that is being addressed. In this case, there are six input nodes and sixteen output nodes for the Portugal data set and six input nodes and eight output nodes for the Scotland data set. Changing the number of input or output nodes implies changing the nature of the problem. For this reason, the only architectural parameters which are varied are the number of hidden layers and the number of nodes in each hidden layer of the network. The influence of the number of input and output nodes on classification accuracy was not investigated. Only networks with one hidden layer were studied. Networks with two hidden layers have not been shown to produce a large improvement in classification results, and they greatly increase the complexity, and thus the training time, of the network (Hepner *et al.*, 1990).

In this set of experiments, the number of nodes in the hidden layer was varied between 1 and 50 for Portugal and between 1 and 30 for Scotland. Fewer network

experiments were run for the Scotland data set because it contains fewer pixels, fewer and more spectrally homogeneous classes, and is therefore a less complex problem requiring fewer hidden nodes. The results obtained in the previous experiments tend to confirm this since the accuracy of classification of the testing set was higher for Scotland than it was for Portugal, even though the former only contained fifteen nodes in its hidden layer whereas the latter had twenty-eight.

Each network architecture was trained once for 600 iterations. The final testing accuracy for each network was used to evaluate the performance of the different architectures by plotting it against the number of nodes in the hidden layer. It was assumed that the final testing accuracy for each run was representative of the maximum accuracy which could be reached for that architecture. In other words, it was assumed that the temporary accuracies were lower for fewer iterations. As seen in the previous experiment, this assumption is reasonable once the network learning has stabilised. Other parameters were kept as for the previous experiments; parameter settings are listed in table 5-4.

Parameters	Portugal	Scotland
Number of iterations	600	600
Architecture: Input Nodes	*6*	*6*
Hidden Nodes	*1 to 50*	*1 to 30*
Output Nodes	*16*	*8*
Type of weight initialisation	Constant seed value	Constant seed value
Learning rate	0.1	0.1
Randomisation	Once	Once
Division ratio	TR : TS = 2 : 1 = 8,047 : 4,458	TR : TS = 2 : 1 = 1,820 : 909

Table 5-4. Parameter settings for the experiment testing the effect of the network architecture on classification accuracy

5.3.2 Results and analysis

Figure 5-3 shows the final accuracy of classification of the testing set for each network with the number of hidden nodes as shown. At first, the accuracy increases as expected. Then beyond a certain minimum threshold, around five hidden nodes for Scotland and fifteen hidden nodes for Portugal, the accuracy remains relatively constant although values oscillate by approximately 5% points for the Scotland data set and by about 7% points for the Portugal data set. There appears to be no pattern in the oscillation, for example with regard to odd and even numbers of nodes, which could suggest a method for determining the best architecture

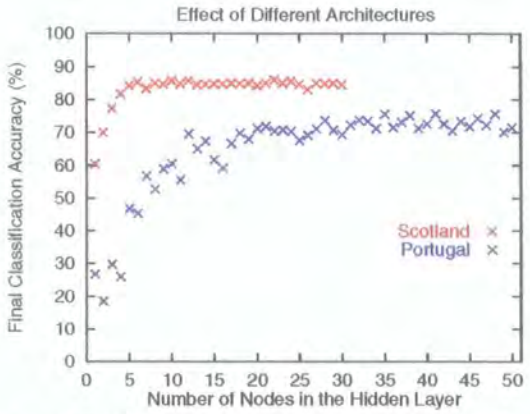


Figure 5-3. Effect of the number of nodes in the hidden layer on testing accuracy

5.3.3 Summary

The testing accuracy curve plotted for the architecture experiments was expected to show a rise, and then a decline, and then stabilisation as networks learned to classify correctly and then over-learned the training patterns. The results show that the accuracy of classification is indeed poor if too few nodes are provided in the hidden layer, then the accuracy rises and stabilises. However, there is no decrease within the range of hidden layer nodes tested in these experiments. This implies that the problem of overfitting may only arise in extreme cases. This corroborates the results obtained in the previous experiment, where testing accuracy did not diminish markedly within the range tested.

Paola and Schowengerdt (1997) tested their network between 1 and 36 nodes in the hidden layer, for a problem involving 12 classes. They report the best accuracy for 9 nodes, when trained to 20,000 iterations, and 12 nodes, when trained to 50,000 iterations, but emphasize the small amount of variability beyond 3 nodes. Skidmore *et al.* (1997) compared the performance of networks with one to three hidden layers. The authors average the results for 1-10 nodes, 11-20 nodes and so on until 41-50. A larger number of layers provide higher accuracies. In their results, accuracy generally varies little within a layer. It increases between 11-20 nodes and then decreases beyond that. Overall accuracy of classification of the testing set is low, averaging 40%.

Considering that large numbers of nodes increase the training time, architectures should be chosen so that a high level of accuracy is obtained but training time is reduced. Therefore, the number of nodes in the hidden layer should be as small as possible without compromising the accuracy. Once the overall accuracy has stabilised, any number of nodes can be chosen. The architecture for the remaining experiments on this Portugal data was set at twenty-eight nodes in the hidden layer; that for Scotland was set to ten nodes.

5.4 Initial Weight Values

As explained in chapter III, section 3.2.2, the network learns by changing its weight values so as to decrease the system error for the next iteration of the pattern. Initial weights should be initialised to small random values since the error is proportional to the weight changes. If weights are too large, then they will require large changes and training is unlikely to succeed. If weights are initialised to the same values, weight changes will be the same and learning cannot take place. The initial weight values therefore influence the final outcome of a classification. However, for a given problem, different sets of random initial weight values should produce similar final accuracies. The aim of this experiment was to determine whether networks with the same configuration but initialised with different random weights would produce different overall accuracies of classification.

5.4.1 Methodology

Network weights for fifty networks were initialised with the current time, as provided by the operating system of the computer, as a seed value. In this way, the configuration of the network is kept constant except for the initial weight values which change for each network. Parameter settings are listed in table 5-5. Results are displayed in a graph showing the final overall testing accuracy for each network and used to evaluate the behaviour of the network implementation.

Parameters	Portugal	Scotland
Number of iterations	600	600
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 28 16	6 10 8
Type of weight initialisation	*Time as a seed;* *50 initialisations*	*Time as a seed;* *50 initialisations*
Learning rate	0.1	0.1
Randomisation	Once	Once
Division ratio	TR : TS = 2 : 1 = 8,047 : 4,458	TR : TS = 2 : 1 = 1,820 : 909

Table 5-5. Parameter settings for the experiment testing the effect of the initial values of the weights on classification accuracy

5.4.2 Results and analysis

Figure 5-4 is a plot of the final testing accuracy after 600 iterations, plotted for each weight case. The networks trained with the data from Portugal demonstrate greater

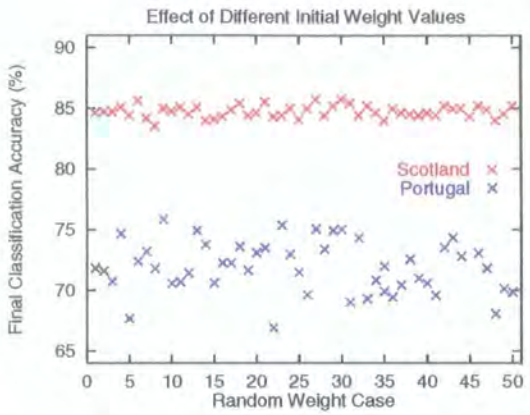


Figure 5-4. Effect of initial value of weight values

differences in their final test data classification accuracies than the networks trained with the Scotland data. Classification percentages oscillate about a mean by about 3% points for the Scotland data set and 8% points for the Portugal data set. There are differences in the overall classification accuracies for each weight case but there is no consistent pattern.

5.4.3 Summary

Networks initialised with different weight values were expected to produce similar final accuracies of classification for the same problem. In fact, the standard deviation from the mean accuracy was about 4% for the Portugal data and 1.5% for the Scotland data; again, it seems that the complexity of a data set has an effect on classification accuracy. These results imply that it is necessary, when a classification accuracy is reported, to average the outcome of several runs with different weight initialisation. The results reported should include mean accuracies and standard deviations for each set of conditions (Paola and Schowengerdt, 1994; Flexer, 1995). However, if the accuracy only serves as an indication of the performance of the network and changes to its value within the standard deviation are not important, one experiment is enough. In this thesis, weights were usually initialised with a constant seed rather than a time seed so that experiments could be re-run and checked. Reported accuracies are only taken as an indication of the average level of performance which can be achieved.

Skidmore *et al.* (1997) report large differences in results and visually different maps for different weight values. They suggest that the differences are a function of the uncertainty associated with each class. The implementation of Paola and Schowengerdt's (1997) network provides different weight initialisations each time. They use the mean accuracy from multiple runs (at least seven) for each of the experiments.

5.5 Learning Rate

The learning rate is a constant, set by the user, which governs the size of changes from one weight value to another during training. Training with small learning rates is slow but the network is expected to converge eventually, as long as it does not get trapped in a local minimum. Large learning rates enable the network to step out of local minima but may cause oscillations so that the network cannot converge (see chapter III, section 3.3.3). The aim of this set of experiments is to determine the effect that different learning rates have on the overall testing accuracy of a network during training. The behaviour of the network during training is studied by plotting the temporary overall testing accuracies at every fifteen iterations. The final overall accuracies are reached at the last iteration.

5.5.1 Methodology

The complexity and size of the data set from Portugal are such that it takes a very long time to train networks. Furthermore, it has been shown in the previous sections that the behaviour of networks trained on the Scotland data set are similar to those of networks trained on the Portugal data set, although the latter show larger oscillations. Therefore, for this and the next set of experiments, networks were only trained and tested on the data from Scotland. Eleven networks with the parameters listed in table 5-6 were trained and tested with learning rates in steps of 0.1 between 0.1 and 1.0 and with an additional step of 0.05.

Parameters	Scotland
Number of iterations	600
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 10 8
Type of weight initialisation	Constant seed value
Learning rate	*0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. 1.0*
Randomisation	Once
Division ratio	TR : TS = 2 : 1 = 1,820 : 909

Table 5-6. Parameter settings for the experiment testing the effect of the learning rate on classification accuracy

5.5.2 Results and analysis

The results are illustrated in figure 5-5.a and figure 5-5.b which show the temporary overall testing accuracies, for each network trained with a different learning rate, at every iteration. The graphs show that small learning rates (figure 5-5.a) provide a stable learning

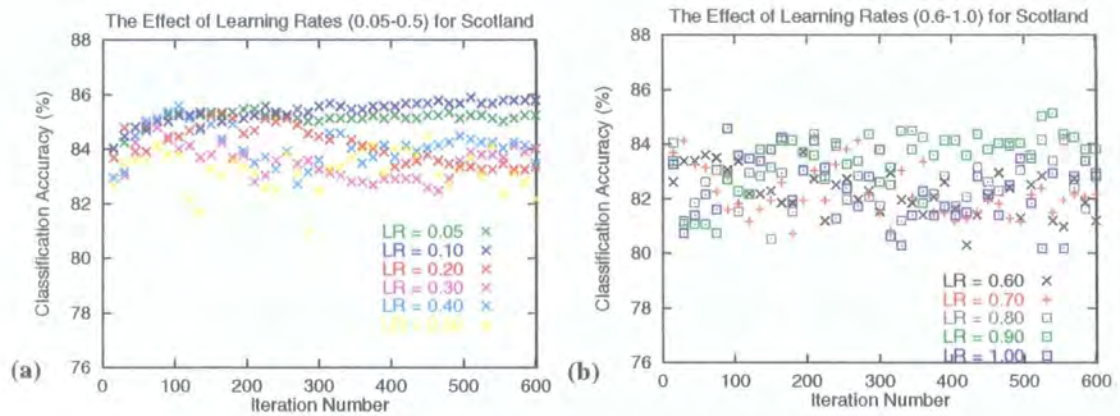


Figure 5-5. Effect of learning rate on classification accuracy. (a) Learning Rates 0.1-0.5; (b) Learning Rates 0.6 - 1.0

cycle with an overall testing accuracy which increases smoothly with increasing numbers of iterations. On the other hand, large learning rates (figure 5-5.b) cause oscillations in the overall testing accuracies which show no particular pattern with increasing numbers of iterations. Furthermore, the final testing accuracies reached with low learning rates are higher than with high learning rates. The learning rate of 0.05 is an exception as it does not consistently produce higher accuracies that the learning rate of 0.1

5.5.3 Summary

As expected, small learning rates yield more stable learning than larger learning rates which produce oscillations in the network. Furthermore, small learning rates produce higher final accuracies than larger learning rates except for 0.05. For these reasons, the learning rate was set at 0.1 for all subsequent experiments.

Skidmore *et al.* (1997) investigated the influence of learning rates coupled with momentum rates on the network performance. They conclude that momentum rates accelerate learning but have no effect on classification accuracy and high learning rates produce lower classification accuracies; the latter agrees with the findings of this set of experiments. The authors also study the number of iterations required to achieve the same accuracy and find that low learning rates require more iterations. This is contrary to the experiments reported here which have shown that small learning rates give overall higher accuracies almost immediately.

A common procedure is to observe the error of the system and change the learning rate accordingly since in training the error may stagnate after a while, particularly with small learning rates. For example, Kanellopoulos *et al.* (1994a) suggest decreasing the learning rate if the error has remained approximately constant for ten iterations. Paola and

Schowengerdt (1995b) compare the n^{th} error with the $n\text{-}1^{th}$ error at fixed intervals. If the former is larger then the learning rate is decreased, otherwise it is increased. The advantage of such a method is that it places little importance on the original values set by the user and convergence is more likely to occur. Experiments were run where the learning rate was diminished by 0.02 when the error had changed by less than 0.1 in ten iterations but there was no improvement in the accuracy of classification for these data. This procedure was therefore not carried out further in the thesis.

5.6 Composition of the Data Sets

The testing data should be representative of the training data for an unbiased estimation of the capabilities of the network (see chapter III, section 3.3.5 and section 3.3.6). The question arises as to whether the same data set produces different classification accuracies if the data are presented to the network in a different order, through different randomisations, and/or if the number of training pixels to the number of testing pixels ratio changes. As long as patterns are randomly presented to the network and not class by class, since this may cause the network to be biased towards the last set of patterns, the final accuracy should not be affected. On the other hand, fewer pixels in the training data set may make it less representative of the testing data set. In this case, final test data classification accuracies may decrease with lower ratios of training to testing pixels. The aim of this set of experiments was to determine whether different randomisations of the same data would produce different classification accuracies, and also to determine whether decreasing ratios of training to testing pixels would produce a reduce level of testing accuracy.

5.6.1 Methodology

The influence of different randomisations of the data files can be evaluated from a plot of final overall testing accuracy for each network trained with a differently randomised set of training and testing files. The accuracy should remain the same if there is no effect. Similarly, the influence of training to testing pixels ratio can be evaluated from a plot of overall testing accuracy for each network trained with differently randomised sets of training and testing files. The 2:1 ratio was tested on thirty differently randomised files; other ratios were randomised ten times.

Networks were trained and tested with the parameter settings listed in table 5-7. 'Randomised 30 times; 2:1' means that 30 different randomisations of the data set were carried out; then each randomised file was divided into a ratio of 2:1 for the number of

pixels in the training file to the number of pixels in the testing file. Each pair of files was then used to train and test a network.

Parameters	Scotland
Number of iterations	600
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 10 8
Type of weight initialisation	Constant seed value
Learning rate	0.1
Division ratio (Training file to Testing file) and Randomisation	*Randomised 30 times; 2:1 - Randomised 10 times; 1:1* *Randomised 10 times; 1:2 - Randomised 10 times; 1:3* *Randomised 10 times; 1:4 - Randomised 10 times; 1:5*

Table 5-7. Parameter settings for the experiment testing the effect of different randomisations and different training pixels to testing pixels ratios on classification accuracy

5.6.2 Results and analysis

Figure 5-6 shows the final classification accuracy for each randomisation case and each division ratio. All the results lie between ~80% and ~85% which shows that the different ratios and randomisations have a limited effect on overall classification accuracy.

Since the division ratio with thirty randomisations showed the same variation as within the first ten randomisations, subsequent division ratios were only tested for ten randomisations. There are no consistent patterns in the randomisation. The division ratios show a slight trend for accuracies to decrease as the ratio decreases. For example, the average accuracy for the 1:5 ratio is lower than for the 1:1 ratio. However, the ratio of 2:1, which should produce higher classification accuracies, does not do so; the accuracies of the networks trained with this ratio are similar to those of the networks trained with a 1:2

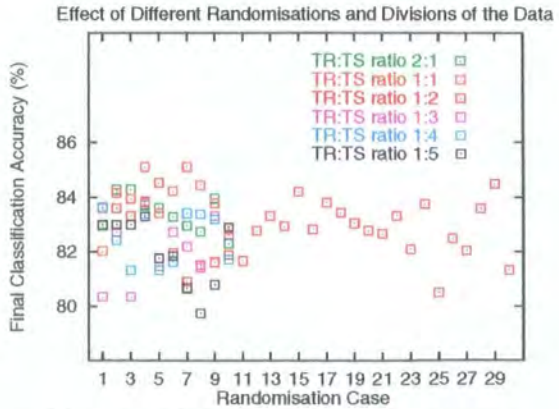


Figure 5-6. Effect of different randomisations and division ratios of the data

ratio. This could be because the network has more patterns to learn the mapping of input to output for the training data which then causes a loss in its ability to generalise.

5.6.3 Summary

The final test data classification accuracies for networks trained with the same data sets, but randomised differently, were not expected to change. However, accuracies were expected to decrease as the ratio of training to testing pixels decreased.

The results of the experiments tend to reinforce these expectations. Accuracies varied by a maximum of 5% points and decreased as the division ratio of training to testing pixels decreased. An exception was the ratio of 2:1 which produced lower accuracies than expected. Considering, as always, that length of training time is an issue, it appears as a rule of thumb that a ratio of 1:2 between number of training patterns and testing patterns is adequate. Furthermore, it seems possible to assume that the level of accuracy produced for any one experiment will be almost the same as the level of accuracy produced by a network with the same configuration but differently randomised data.

Skidmore *et al.* (1997) report on experiments to test whether a network tested on randomised data performs better than on sequential data. Unusually, they report higher accuracies with the sequential data. However, they comment that even the sequential data may be quite random as the data set is large. Since details of the methodology and contents of the data sets are not provided in the article, it is not possible to draw general conclusions. They do not test different ratios or different randomisations. In an earlier set of experiments, Paola and Schowengerdt (1995a) find that training with small data sets produce larger training accuracies but lower testing accuracies. Staufer and Fischer (1997) measure the overall accuracy of classification of a network and find that when the training data is diminished by 40% the overall accuracy only changes by -1.24%, and when the training data is diminished by 80% the accuracy decreases by 6.66%.

5.7 Summary of Chapter V

The aim of the experiments in this chapter was to investigate the effect that changes to different parameters of the network may have on the accuracy of classification. Each parameter was tested in turn, keeping all others constant.

In section 5.2, the classification accuracy was shown to increase with the number of iterations of the data until a point beyond which it stabilises and may eventually decrease. The error of the system decreases with the number of iterations until a point beyond which

it stabilises and may eventually increase. Since the error is measured on the training data set, it is not used subsequently as a measure of the performance of the network. The number of iterations to which networks will be trained for further experiments is set to 600. It is understood that higher accuracies may be reached with more iterations but 600 should produce a representative accuracy.

Experiments concerned with the architecture of the network, and more particularly, the number of nodes in the hidden layer, showed that the classification accuracy increases with the number of hidden nodes until a point beyond which it stabilises (section 5.3). For these data sets, the best architecture was 28 nodes in the hidden layer for the Portugal data and 10 for the Scotland data. Subsequent experiments which use similar architectures for similar data sets are expected to produce accuracies which are representative of the best accuracy, even if they may not be the best architecture.

Different initial weight values were studied in section 5.4 and shown to produce classification accuracies which oscillate within a small range dependent on the problem. The oscillation of the accuracies for different initial weight values is expected to be small enough that reporting the results of only one network will be representative of the results which would be obtained when training with several different weight initialisations.

Learning rates were investigated in section 5.5. Small learning rates generally produce smoother learning curves and higher final classification accuracies than larger learning rate. A learning rate for subsequent experiments of 0.1 is expected to produce smoother learning and higher classification accuracies than larger learning rates.

Finally, characteristics of the training and testing data set were examined in section 5.6. As the division ratio of the training to testing pixels increases, the final classification accuracy tends to increase. The data sets used in subsequent experiments are divided in a ratio of 1:2 for the number of pixels of training data to the number of pixels of testing data. This division is expected to provide an adequate generalisation in less time than a higher division ratio.

Presenting the network with differently randomised data sets produces classification accuracies which oscillate within a small range dependent on the problem. Results from networks trained with a single randomised data set are expected to be representative of the results that would be obtained from a set of networks trained with different randomisations.

Oscillation effects are expected to cancel each other out since they appear to be randomly distributed. For example, if a particular set of initial weights produces a classification accuracy which is lower than the mean classification accuracy (achieved

with several initial weight values), this effect will be minimised or cancelled if the particular randomisation of the data set produces a classification accuracy which is higher than the mean classification accuracy (achieved with different randomisations).

The results generally agree with those presented in the literature such as by Paola and Schowengerdt (1997) and Skidmore *et al.* (1997). Skidmore *et al.* (1997) argue that the choice of network parameters is critical to success. From the results in this chapter, it is suggested that if appropriate values are selected for the parameters, then results will average to the same overall accuracies. Particularly if the best performance is not of interest, the user can run the network without having to worry that choices that have been made will have a significant consequence on the results. On the other hand, if the best performance is of interest, then a systematic review of parameter values and average overall accuracies must be undertaken. Having investigated the sensitivity of the network, the study of the neural network outputs can now take place.

Chapter VI

Qualitative Analysis Tools and Sub-Pixel Information

In the previous chapter the effect of changes to the parameter settings of the neural network was assessed. The study of neural network outputs can take place with the confidence that appropriate choices of iteration number, architecture, initial weight values, learning rate and training and testing file division ratio and randomisation do not unduly influence observations. A combination of qualitative and quantitative analyses are used to examine neural network representation of mixed pixels. This chapter provides a qualitative appraisal of the neural network outputs.

The first section of this chapter investigates the relationship between neural network output values and correctly or incorrectly classified pixels for a data set containing only *pure* pixels. This serves to illustrate the reasons which led to the initial hypothesis that neural network output values may reflect the proportions of classes present within pixels. As discussed in chapter II, several authors have suggested that there is a relationship between neural network output values and percentage cover of classes within pixels (Atkinson *et al.*, 1997; Foody *et al.*, 1997; Moody *et al.*, 1996) and the remainder of the chapter describes experiments carried out to test this hypothesis.

Subsequent sections use visual tools to analyse the results of *soft* classification by the neural network. Fraction images, described in the second section, are similar to those normally produced in linear un-mixing methodologies and allow a spatial analysis of neural network output values. However, the relationship between classes within a pixel is lost using this type of output. A new non-spatial visualisation tool is described in the third section that allows the direct comparison of reference data and neural network outputs and keeps the relationship among classes intact. The fourth and final section examines the representation of mixed pixel data by the neural network, particularly for unknown classes, using a new graphical technique which shows the network response at every output node for pixels grouped according to their dominant class.

6.1 Network Output Strength Statistics

The aim of the experiments reported in this section is to determine whether there is a relationship between output values and correctly or incorrectly classified pixels in a *pure* data set. Although a data set is considered to contain only *pure pixels,* it may in fact contain some mixed pixels. A pixel may be incorrectly classified by the network if its spectral signature is similar to the spectral signatures of more than one class or if its spectral signature is unlike the spectral signature of its dominant class. This may happen if classes are not spectrally separable or the pixel is mixed. If there is a difference between the response of the network for correctly classified pixels and incorrectly classified pixels, this would suggest that neural networks may be able to identify mixed pixels and perhaps their component classes.

As discussed in chapter II, there is mathematical proof that outputs from a multi-layer perceptron, when they are scaled between zero and one, approximate Bayesian *a-posteriori* probabilities under a series of conditions which include infinitely large data sets, adequate architecture and correct representation of the true data statistics by the training data (Richard and Lippman, 1991; Ruck *et al.*, 1990; Shoemaker, 1991). If this is so, then scaled output values from the neural network should sum to one (Lowe and Webb, 1991). It is also presumed that if neural network outputs are *a-posteriori* probabilities, correctly classified pixels will have a high value in the dominant class node and low values elsewhere. Kanellopoulos and Wilkinson (1990) found that the average value of maximal firing of the output units was substantially higher for correctly classified pixels than for misclassification cases. On the other hand, misclassified pixels are expected to have high values in more than one output node, corresponding to the classes that are spectrally similar or that are present within the pixel.

6.1.1 Methodology

The following hypotheses were tested using Mann-Whitney U tests, graphs and mean and standard deviation calculations.

1- the sum of the output strengths for correctly and incorrectly classified pixels is equal to one;

2- the maximum output for correctly classified pixels is greater than the maximum output for incorrectly classified pixels;

3- the second maximum output for correctly classified pixels is smaller than the second maximum output for incorrectly classified pixels;

4- the difference between the maximum output and the second maximum output is greater for correctly classified pixels than for incorrectly classified pixels.

The Mann-Whitney U test is similar to the student t-test except that the latter assumes that data is normally distributed and is usually applied to small data sets. The Mann-Whitney U test assumes similar distributions between data sets with possibly different medians. Skidmore *et al.* (1997) use this test for their investigation of the effect of changes to neural network parameters. Mean and standard deviation statistics and the Mann-Whitney U test provide a summary statistic for the whole data set. On the other hand, plots of individual pixel values show the distribution of results for each pixel. Therefore, the hypotheses are tested using a combination of these techniques.

Neural network experiments were run on the *pure* 'Portugal sixteen class' data set using the parameters listed in table 6-1.

Parameters	Portugal 16 class
Number of iterations	600
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 28 16
Type of weight initialization	Constant seed value
Learning rate	0.1
Randomisation	Once
Division ratio	TR : TS = 2 : 1 = 8,047 : 4,458

Table 6-1. Parameter settings for the 'Portugal sixteen class' data set

6.1.2 Results and analysis

The final accuracy of classification was approximately 75%. The network output values from the output file created at the testing stage were scaled between 0 and 1 and the following variables were identified or calculated for each pixel: the sum of the scaled outputs, the maximum value of the outputs, the next (or second) maximum value of the outputs and the difference between the latter two. The testing data was then divided into two groups: correctly classified pixels and incorrectly classified pixels. There were 3,292 correctly classified pixels and 1,166 incorrectly classified pixels in the test set. Mean and standard deviations were calculated for each variable within each group. Results are recorded in table 6-2.

The table shows that the mean values of the sum of the scaled outputs are close to one but not exactly one. Incorrectly classified pixels might have been expected to have a mean that is further from the theoretically ideal sum of one but this is not the case; the

	3,292 Correctly Classified Pixels		1,166 Incorrectly Classified Pixels	
Variable Name	**Mean**	**Standard Deviation**	**Mean**	**Standard Deviation**
Sum of outputs	1.17	0.28	1.14	0.37
First maximum	0.88	0.17	0.70	0.24
Second maximum	0.18	0.15	0.28	0.19
Difference between the first and second maximums	0.71	0.25	0.41	0.28

Table 6-2. Statistics from the neural network outputs of the 'Portugal sixteen class' data set

mean of the correctly classified pixels is further from one than the mean for the incorrectly classified pixels. On the other hand, the standard deviation for the incorrectly classified pixels is larger than for the correctly classified pixels.

The mean of the maximum value for correctly classified pixels is higher than for the incorrectly classified pixels. If neural network outputs do indeed represent the *a-posteriori* probability of assigning pixels to each class as suggested in the literature, then these results make sense. Incorrectly classified pixels are likely to have a spectral signature that is similar to more than one class, the network therefore has difficulty in assigning any of the classes with any certainty.

The mean second maximum output value is lower for correctly classified pixels than it is for incorrectly classified pixels. Again, this suggests that for correctly classified pixels the network has some degree of certainty that the spectral signature is not similar to any of the other classes. On the other hand, for incorrectly classified pixels, the network is less certain that the pixel may not belong to other classes. The mean difference between the mean maximum output value and the mean next maximum output value is consequently larger for correctly classified pixels than it is for incorrectly classified pixels.

The hypotheses concerning the difference between correctly classified and incorrectly classified pixels, hypotheses 2, 3 and 4, were tested formally with a Mann-Whitney U test using S^+ software. Stated formally, the second hypothesis tests the null hypothesis H_0: $n_1 = n_2$ versus the alternate hypothesis H_1: $n_1 > n_2$ where n_1 is the median first maximum for correctly classified pixels and n_2 is the median first maximum for incorrectly classified pixels. The null hypothesis was rejected at $p < 0.0001$ confirming that the correctly classified pixels have a higher first maximum than incorrectly classified

pixels. Unfortunately, when values are very close to zero or very close to one, the S^+ software does not report exact values but rather rounds the numbers to zero and one.

The third hypothesis tests the null hypothesis H_0: $n_1 = n_2$ versus the alternate hypothesis H_1: $n_1 > n_2$ where n_1 is the median first maximum for correctly classified pixels and n_2 is the median first maximum for incorrectly classified pixels. The null hypothesis was rejected at $p < 0.0001$ confirming that the correctly classified pixels have a higher first maximum than incorrectly classified pixels.

The fourth hypothesis tests the null hypothesis H_0: $n_1 = n_2$ versus the alternate hypothesis H_1: $n_1 > n_2$ where n_1 is the median difference between the first and second maximums for correctly classified pixels and n_2 is the median difference between the first and second maximum for incorrectly classified pixels. The null hypothesis was rejected at $p < 0.0001$ confirming that the correctly classified pixels have a larger difference between the first and second maximums than incorrectly classified pixels.

The summary statistics that have been discussed provide overall accuracy values that can be compared but they provide no information regarding the distribution of results over the range of data values. Graphs of the values for each pixel can show the distribution of results. Figure 6-1 contains plots of the sum of the scaled outputs, figure 6-1.a, the first maximum, figure 6-1.b, the second maximum, figure 6-1.c and the difference between these two, figure 6-1.d for each pixel in the test set. Correctly classified pixels are plotted in green; incorrectly classified pixels are in plotted in blue.

Figure 6-1.a shows that there appears to be no pattern between correctly and incorrectly classified pixels in terms of the sum of their outputs. As mentioned above, misclassified pixels might have been expected to have consistently lower or higher sums of outputs than correctly classified pixels but this does not appear to be the case. The incorrectly classified pixels tend to be grouped together. Since the testing file had not been randomised, neighbouring pixels belong to the same class. This shows that some classes are consistently misclassified, which may mean that they are either poorly defined or they contain many mixed pixels, whereas others are consistently correctly classified.

Figure 6-1.b shows that incorrectly classified pixels rarely reach a maximum output value of 1 unlike the correctly classified pixels, and figure 6-1.c shows that their second maximum is rarely in the region of 0. Pixels which have a maximum close to 1 often have a second maximum close to 0. In other words, when the neural network is certain about an assignment it seems to be equally certain about not assigning any of the other classes and the difference between the first and the second maximum is large as shown in , figure 6-1.d.

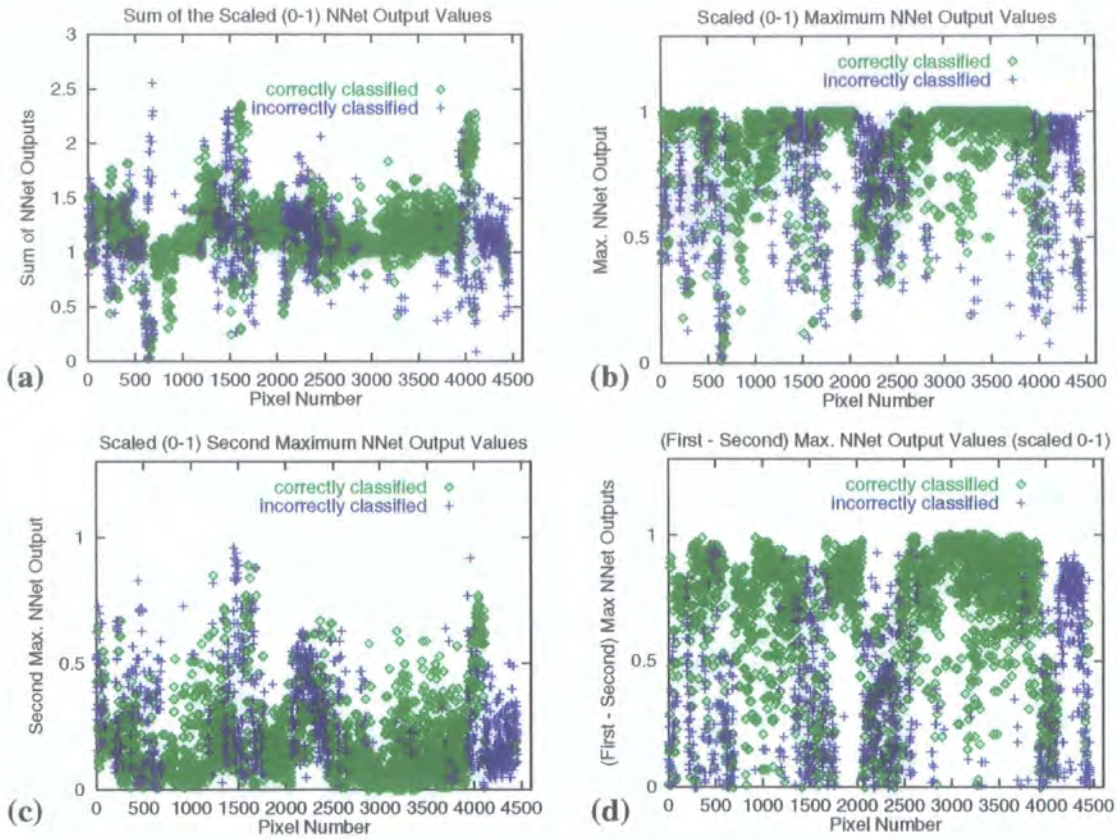


Figure 6-1. (a) sum of the neural network outputs; (b) maximum neural network output values; (c) second maximum neural network output values; (d) difference between the first and second neural network output values

A similar experiment was carried out on the 'Portugal fifteen class' data set to examine the response of the neural network when using a data set that is known to contain a high proportion of mixed pixels. A neural network was used with parameters set to those listed in table 6-3.

Parameters	Portugal 15 class
Number of iterations	600
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 28 15
Type of weight initialization	Constant seed value
Learning rate	0.1
Randomisation	Once
Division ratio	TR : TS = 2 : 1 = 9,217 : 18,434

Table 6-3. Parameter settings for the 'Portugal fifteen class' data set

The mixed pixels were treated as *pure* in this experiment in terms of the target which was assigned to them. The dominant class within the pixel was assigned +1, all the others

were assigned -1. For example, the target for a pixel with a composition of [30% Class 1 + 50% Class 6 + 20% Class 10] would be +1 for class 6 and -1 for all other classes. Class labels and their percentage cover values are read into vector arrays in the order in which they are presented in the reference data file. For compositions with two or more dominant classes, the last dominant class in the vector array is set to +1, the others are set to -1. For example, a composition of [30% Class 10 + 20% Class 7 + 30% Class 6 + 20% Class 15] would be represented by a target with a value of +1 for class 6 and -1 for all other classes. These pixels could introduce some negative bias in the accuracy of classification since a pixel may be identified as misclassified even though it is correctly classified. In the example above, identifying the pixel as belonging to class 10 is in fact as correct as identifying it as belonging to class 06. There are 5,257 pixels with two or more classes with equal cover in the 'Portugal fifteen class' data set, which corresponds to almost 20% of the data set. However, many of these pixels have one dominant class and then equal proportions of second and third classes.

Classification accuracy of the 'Portugal fifteen class' data set is similar to that of the 'Portugal sixteen class' data set: ~ 70%. There were 12,880 correctly classified pixels and 5,554 incorrectly classified pixels. The mean sum of outputs, mean maximum output values, mean second maximum output values and the mean difference between the latter two for each group of pixels are listed in table 6-4 together with their standard deviations.

	12,880 Correctly classified pixels		5,554 Incorrectly classified pixels	
Variable Name	**Mean**	**Standard Deviation**	**Mean**	**Standard Deviation**
Sum of outputs	3.62	0.05	3.61	0.05
First maximum	0.16	0.03	0.13	0.03
Second maximum	0.09	0.01	0.10	0.01
Difference between the first and second maximums	0.07	0.04	0.04	0.03

Table 6-4. Statistics from the neural network outputs of the 'Portugal fifteen class' data set

The mean sum of the scaled outputs is much higher than for the previous data set: approximately 3.6 compared to approximately 1.1. The values of the neural network outputs for the maximum and the second maximum, regardless of whether the pixel is classified correctly or not, are much lower than in the previous experiment, see table 6-2. Even though the scaled neural network outputs can take on values as high as 1.0, their mean value is ~0.15. The difference between the two maximums is again smaller for

misclassified pixels than it is for correctly classified pixels, but the size of the difference is very small as are the standard deviations for each variable.

As emphasized in section 6.1, mathematical theory suggests that neural networks approximate Bayesian *a-posteriori* probabilities and output values should consequently sum to one. For a network with an activation function that does not lie between 0 and 1, as is the case here, network outputs must be scaled. Since it is the network's ability to approximate *a-posteriori* values naturally that is of interest, outputs are not normalised (forcing the sum to one by dividing each output value by the sum of the outputs).

In this experiment, the sum of the output values is much larger than one and the network therefore does not appear to approximate *a-posteriori* probabilities. However, comparing the ground data with the resulting output values suggests that they are closely connected. Dominant classes of a pixel are dominant in the network output values and secondary classes often have higher output values that other classes. This suggests that the output values of the network are related to *a-posteriori* probabilities but that the relationship may be complex. Simply linearly scaling the outputs between -1.7 and 1.7 to lie between 0 and 1, for example, may tend to exaggerate the importance of small output values. Further analysis of these results is presented in section 6.2 and section 6.3.

6.1.3 Summary

The aim of the experiments in this section was to determine whether there was a relationship between neural network output values and correctly or incorrectly classified pixels. There appeared to be no relationship between correctly or incorrectly classified pixels and the sum of output values. The mean sum of the outputs for the 'Portugal sixteen class' data set was close to 1.1 whereas the mean sum of the outputs for the 'Portugal fifteen class' data set was close to 3.6. This suggests that the sum of the neural network outputs is not an indication of overall accuracy of classification but may be useful as a measure of the complexity of the data set, and possibly, of the presence of mixed pixels.

The mean output values for the first maximum and second maximum showed a relationship with correctly and incorrectly classified pixels. Correctly classified pixels had larger maximum output values than incorrectly classified pixels. Using a maximum likelihood classifier on a per-field basis for crop analysis, Foody *et al.* (1992) found similar results although the difference between correctly and incorrectly classified pixels was small. Correctly classified pixels had smaller second maximum values and consequently larger differences between the first and second maximum than incorrectly classified pixels.

Assuming that pixels are misclassified because they are either mixed or some classes are spectrally similar, the results imply that the network produces output values for mixed or misclassified pixels that are different from those produced for correctly classified pixels. If this is true, then it may also be the case that neural network output values reflect which classes mix, or misclassify, and perhaps even the proportions of classes in mixtures. However, the fact that the mean sum of the network outputs for the 'Portugal fifteen class' data set was not close to one, indicates that network output values may not be able to approximate *a-posteriori* probabilities in the presence of mixed pixels.

6.2 Neural Network Fraction Images

Fraction images, also called proportion or abundance images, are typically a product of linear un-mixing models (Donoghue, 1994; Gong *et al.*, 1994; Oleson *et al.*, 1995; Smith *et al*, 1990) although they have also been produced from other methodologies (Fisher and Pathirana, 1990; Maselli *et al.*, 1996). For each class, a single band grey scale image is produced where the intensity of each pixel is proportional to the percentage cover of the pixel by the class. Similar images can be produced from neural network outputs (Foody and Arora, 1996) to visualise the spatial distribution of class assignment.

6.2.1 Methodology

The hypothesis that neural network output values can be considered *a-posteriori* probabilities of classification, under ideal conditions, was discussed in chapter II where it was also shown that the basis of most methodologies is that *a-posteriori* probabilities are linked to the percentage cover of a pixel by the class. Therefore, if single class images are created where the intensity of each pixel is proportional to the neural network output values, the images should represent certainty and uncertainty of classification and, by extension, this uncertainty may relate to proportions of each class in a mixture.

The original neural network software was modified so that for each pixel, the output values at each node could be provided instead of only the class to which the pixel is assigned (chapter III, section 3.4). The output file created from the 'Portugal fifteen class' data set classification in the previous experiment, section 6.1, was used here. Since the outputs of the neural network lie between -1.7 and 1.7, they were scaled to lie between zero and one and multiplied by 100 so that an adequate mapping could be produced on the image processing system between the neural network outputs which lie in a range [0,1] and the grey scale values which lie in the range [0,255]. If required, the outputs can be normalised so that they sum to one. The intensity of the pixels in each single class image

is proportional to the neural network output value of the pixel for that class. The procedure for creating these output strength images, or fraction images, is illustrated in figure 6-2.

6.2.2 Results and analysis

Fraction images created from the neural network classification of the Landsat TM scene of the Portugal site are presented in figure 6-3 and figure 6-4 in the form of individual single-class neural network fraction images. The neural network values were mapped to the grey scales so that black to white represented low output values to high output values.

The images for the **Stones** class and the **Fruit Trees** class are almost entirely black with only a dozen or so pixels in white. This implies that either there are almost no pixels within which these classes are present or these classes cannot be recognised by the neural network because they are confused with other classes. Analysis of the ground data from chapter IV revealed that the **Stones** class was only present as 10% cover within a mixture with 90% **Shrubs** and the spectral signature of the **Stones** class was most similar to the **Shrubs** class signature. Therefore, the classifier will not have been provided with a target for the **Stones** class, since a *pure* type of target was created which in this case will, quite rightly, assign the pixels to the **Shrubs** class. Consequently, the classifier cannot be expected to classify this class correctly.

There are 399 pixels containing the **Fruit Trees** class but only 63 of these have it assigned as a dominant class, and therefore have a corresponding **Fruit Trees** target. The spectral signature of **Fruit Trees** is also quite similar to **Shrubs**. Consequently, the network has difficulty classifying the **Fruit Trees** class.

The **Water** class appears to be well defined. From the reference data, there were 1,648 pixels containing **Water** of which 1,292 as dominant class (78%). Areas on the image which should have a water signal, such as the sea, do indeed have a high water signal. Furthermore, these areas generally do not register signals from any other node. The area within the estuary is an exception as it contains **Grass** and **Marsh** signals, suggesting that there is some vegetation in that area. It is interesting that the actual **Marsh** class which can be thought of as a mixture between **Water** and **Grass** is less present, from the grey scale colours, than the **Grass** signal. This could imply that it is more profitable to identify *pure* classes as the basic classes, rather than use already mixed classes such as **Marsh**. On the other hand, the **Marsh** class is present along the rice field 'arms' which stream down from the fresh water river, in the top right of the image. Since the **Marsh** class was created from the amalgamation of original classes 'rice' and 'aquatic plants', it

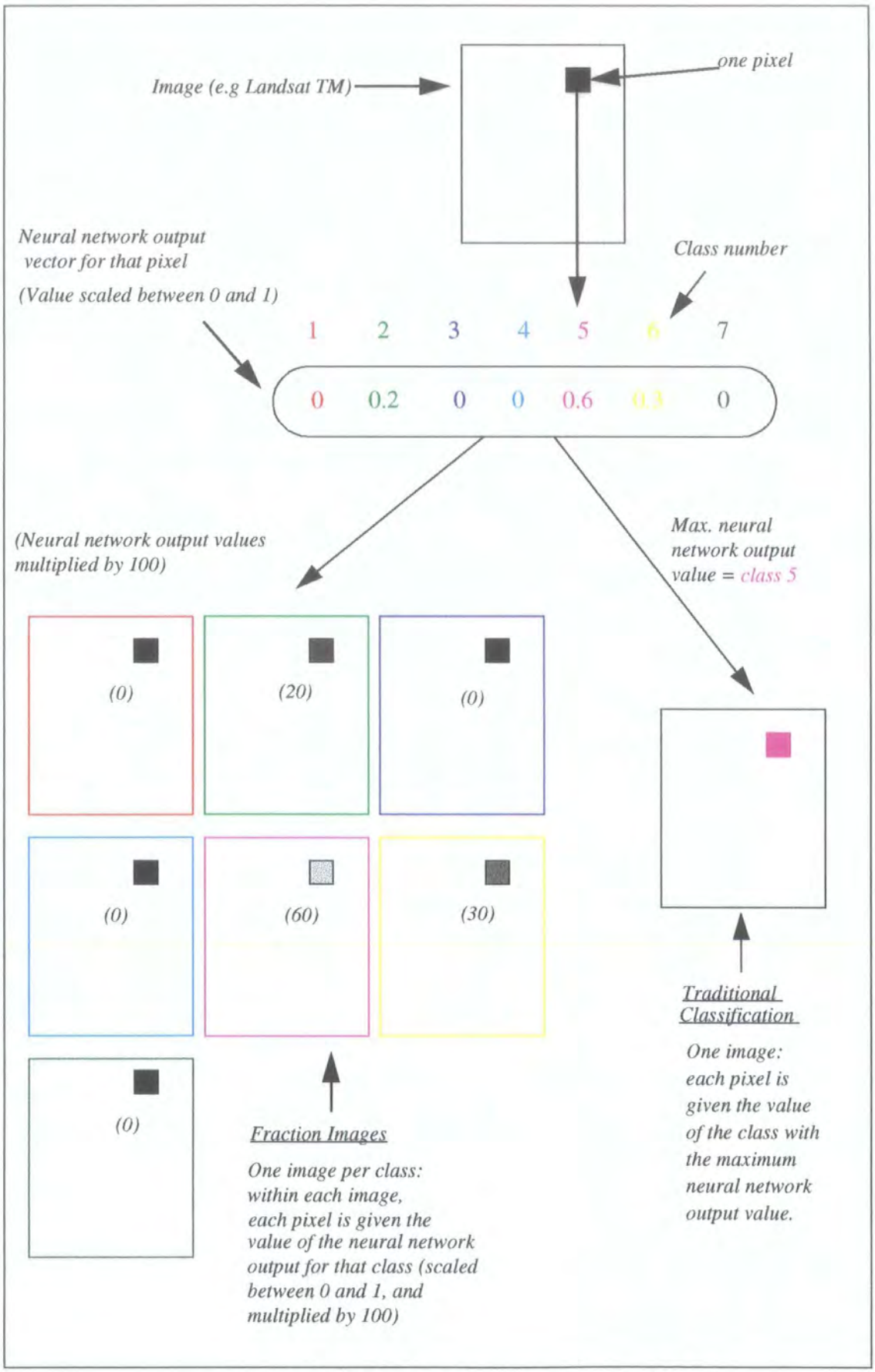


Figure 6-2. Procedure for creating neural network fraction images

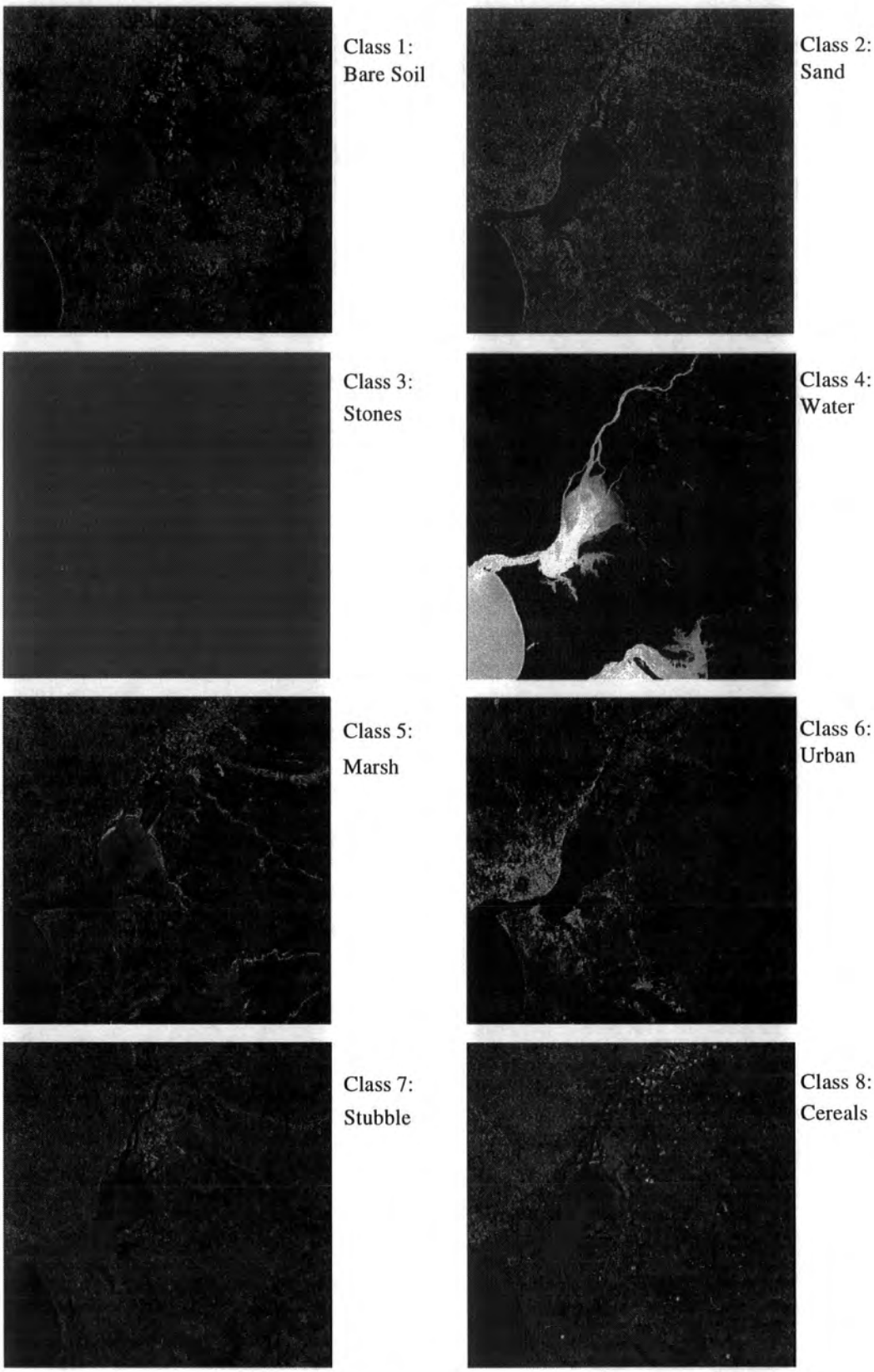


Figure 6-3. Neural network fraction images for classes 1-8 of the 'Portugal fifteen class' data set

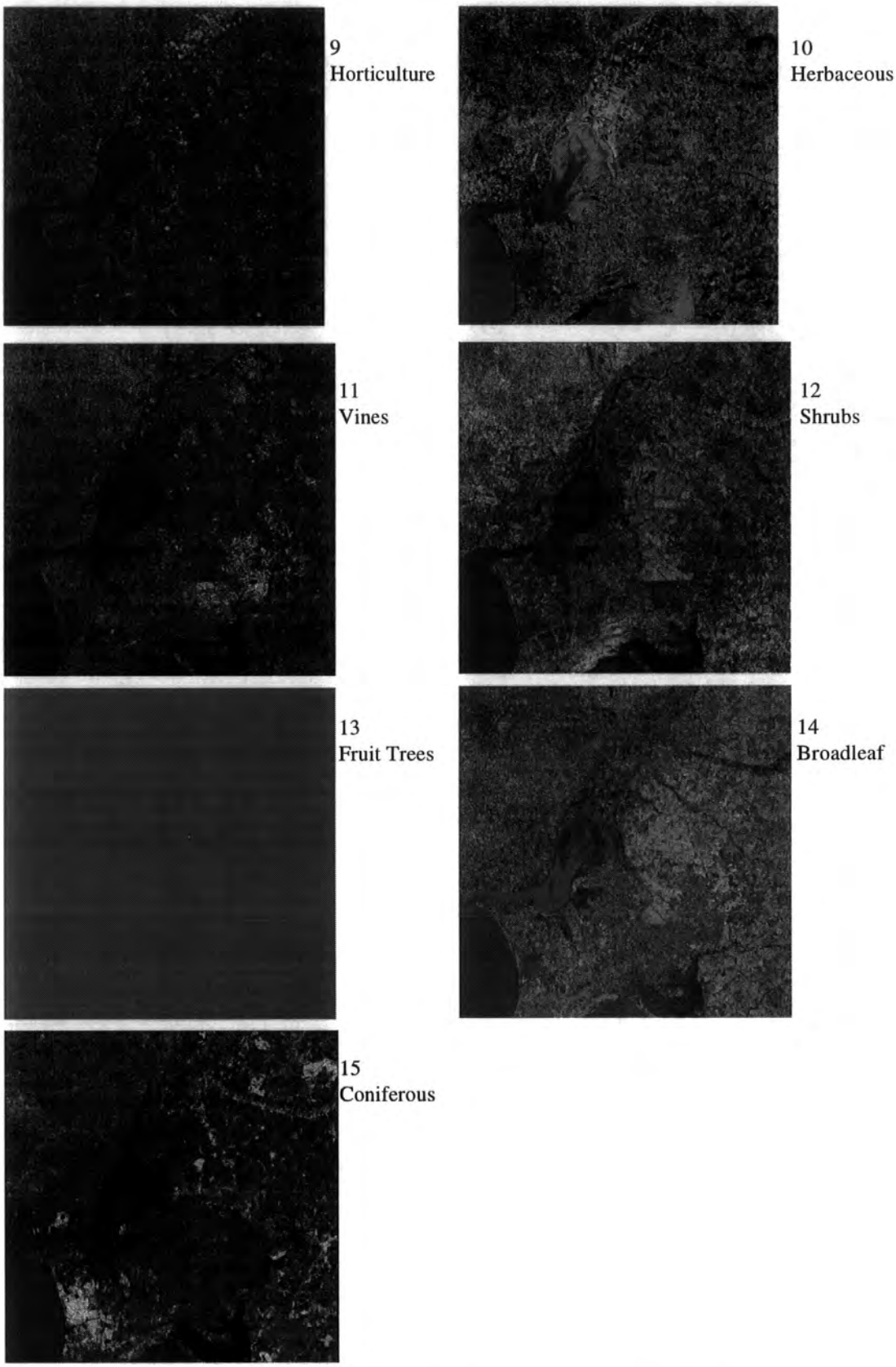


Figure 6-4. Neural network fraction images for classes 9-15 of the 'Portugal fifteen class' data set

may be that the vegetation present in the estuary has a signal which is closer to **Grass** than to 'rice'. However, it is also true that there are only 208 pixels that contain **Marsh** but 17,372 pixels that contain **Grass**.

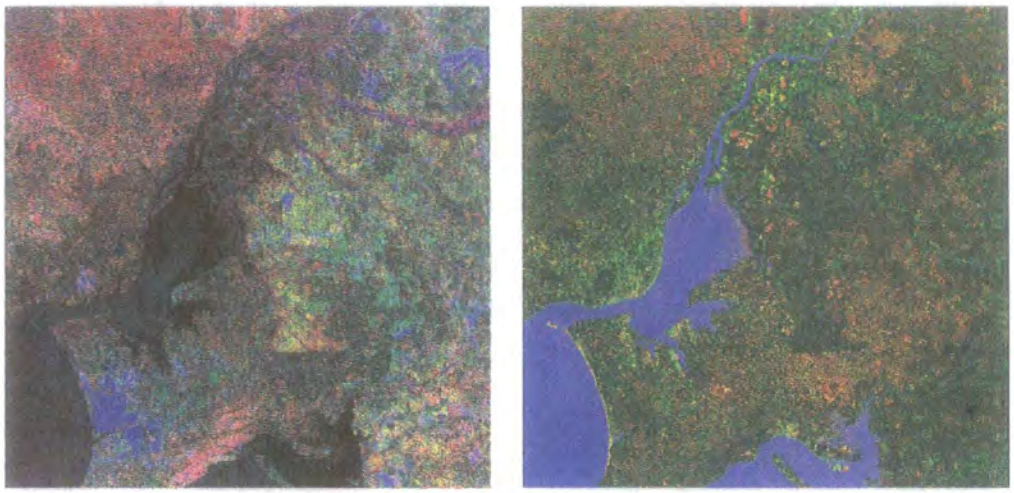
Other fairly well defined classes, well defined in the sense that their occurrence is in well defined areas and not spread out over the whole image, include the **Urban** class and the **Coniferous** class. This is slightly unexpected since their spectral signatures (chapter IV, section 4.4) showed them to contain quite a few outliers.

Many of the remaining classes seem to induce a small signal in all pixels. This can be explained for some classes such as **Grass** or **Bare Soil,** which can be expected to exist in almost every pixel, but other classes such as the **Sand** class are less understandable. It is possible that **Sand**, of which there are only 102 pixels in the data set, and **Bare Soil** classes have been confused. Particularly badly identified classes are **Broadleaf** and **Shrubs**, both of which resemble each other and other classes spectrally, as mentioned in chapter IV.

This analysis technique provides a spatial overview of the distribution of the classes allocated by the network. However, although it is possible to identify the presence and absence of classes in particular areas, it is not possible to obtain details about individual pixels. The quantity or percentage cover of each class is not evident, nor are the relative proportions of classes which appear to mix in some areas. Creating a pseudo colour table so that, for example, strengths 0-10 are coded blue, 11-20 are coded red and so on, does not particularly alleviate the problem.

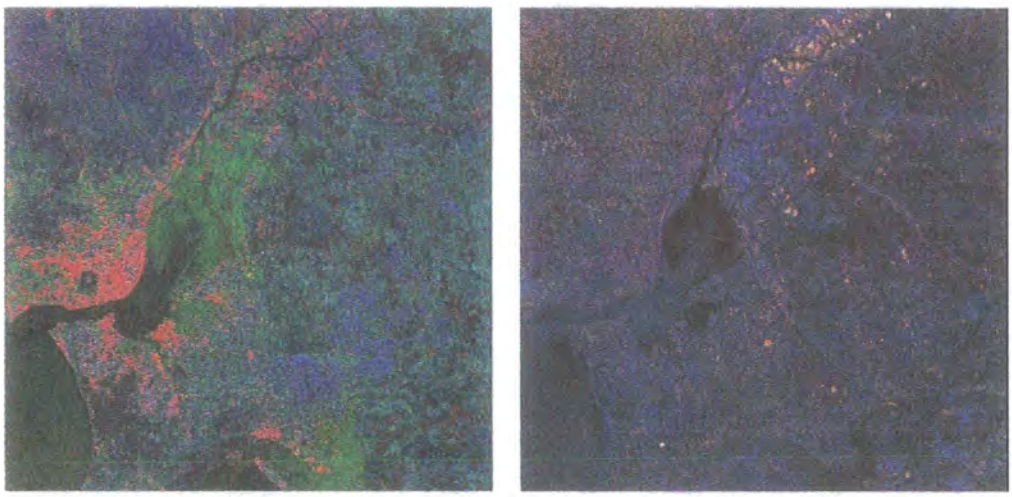
To improve the analysis, and since the eye is more sensitive to colour differences that to grey levels (Adams *et al.*, 1986), it is possible to load three fraction images on to the colour guns (Red, Green, Blue) of an image processing system. In this way, intensity of colour gives an approximate idea of the percentage cover by the class on the colour gun being viewed. When the guns are combined, mixing between classes is identified by mixed colours: purple is produced by a mixture of red and blue; yellow is produced from a mixture of red and green; and cyan is produced from a mixture of blue and green. A similar procedure has been used to study changes in vegetation by loading NDVI images of three different dates on the three different colour guns (Sader and Winne, 1992) and with fraction images from a linear un-mixing model (Settle and Drake, 1993; Holben and Shimabukuro, 1993).

A sample of images are shown in figure 6-5. Each single band image is one of the previously shown black and white images. Three classes were chosen for each image.



(a) R = 12 - Shrubs
G = 14 - Broadleaf
B = 15 - Coniferous

(b) R = 1 - Bare Soil
G = 2 - Sand
B = 4 - Water



(c) R = 6 - Urban
G = 10 - Grass
B = 11 - Vines

(d) R = 9 - Horticulture
G = 8 - Cereals
B = 7 - Stubble

Figure 6-5. Colour fraction images - one class per colour gun - mixtures shown by mixture of colours

In the first image (a), although there are pockets of unique colours, in particular for the **Coniferous** forest and the **Shrubs**, most of the image is covered by colours resulting from a mixture of the three primary colours which suggests that there is a considerable degree of mixing (or misclassification) among the three classes according to the neural network. This confirms the expectations of chapter IV that **Coniferous**, **Deciduous** and **Shrubs** would be confused with each other. On the other hand in the second image (b), it is clear that the **Water** class does not mix with the other classes except along the southern coast of the estuary where the purple colour indicates that it mixes with **Bare Soil**. The **Sand** class, as was highlighted in the black and white images above, is distributed throughout the image, as is the **Bare Soil.** The third image (c) shows little mixing among the classes

Urban, **Grass** and **Vines**. All three classes are relatively well defined and there are few mixed colours. Finally, the almost complete absence of Green in the last image (d) suggests that the **Cereal** class is not present in abundance since the other classes overwhelm the colour of the pixels. In addition, the **Stubble** class, much like the **Sand** class, produces a signal in most pixels.

Although it allows some further analysis, this technique has several shortcomings. Firs, a large number of permutations are necessary to analyse the mixing between any three of the fifteen classes; only four examples were shown here. In addition, although some indication of the relative abundances within mixtures is obtained from the mixture colours, the information is not precise. And finally, this technique is limited to a study of up to only three classes at a time.

6.2.3 Summary

The usual product from linear un-mixing methodologies, fraction images, can be created from neural network outputs. However, although they are more informative than a single *pure* classification image, proportion images only provide an overview of the spatial distribution of classes. Degrees of mixing between classes are not obvious. Furthermore, the information present in the reference data is wasted as the fraction images do not allow the comparison between ground data and results. For these reasons, a new visualisation technique was developed.

6.3 Ground Data/Neural Network *Correspondence* Images

The most common visual method used in the literature to analyse the relationship between ground data and the results from a *soft* classification problem is to plot, for each class, the algorithm output (*a-posteriori* probability, membership value or derived percentage cover and so on), against the actual percentage cover as given by the reference data (Marsh *et al.*, 1980; Foody, 1992; Thomas *et al.*, 1996). The graphs show the relationship between the variables and a correlation value can be provided which quantifies it. Although this method shows the distribution of results over the range of data points, it has the disadvantage that only one class is studied at a time and the relationship between mixtures in the same pixel is lost. For example, it is not possible to tell that class **A** was overestimated and consequently class **B** underestimated for a pixel. An alternative method for representing neural network output values is a two dimensional plot of input values of pixels, for each node, where points are coloured according to the strength of the output value (Warner and Shank, 1997; Moody *et al.*, 1996). This technique shows the

neural network partitioning of the input space but again, does not allow class relationships within pixels to be maintained.

The visualisation tool described in this section was developed to allow the direct comparison between the expected results, as provided by reference data, and the actual results, as provided by the neural network outputs for the testing data set, keeping the relationship between classes within a pixel intact. It is a non-spatial visualisation tool since pixels are not viewed in image form.

6.3.1 Methodology

If a pixel is covered 100% by one class only visually this can be represented by a line of set dimension which is coloured entirely in the colour assigned to the class (figure 6-6 upper bar). If only a portion of the pixel is covered by a class, relative amounts can be coloured accordingly. For example, the lower bar in figure 6-6, could represent a [30% Class 2 + 60% Class 5 + 10% Class 10] mixture, where class 2 is in blue, class 5 is in green and class 10 is in red.

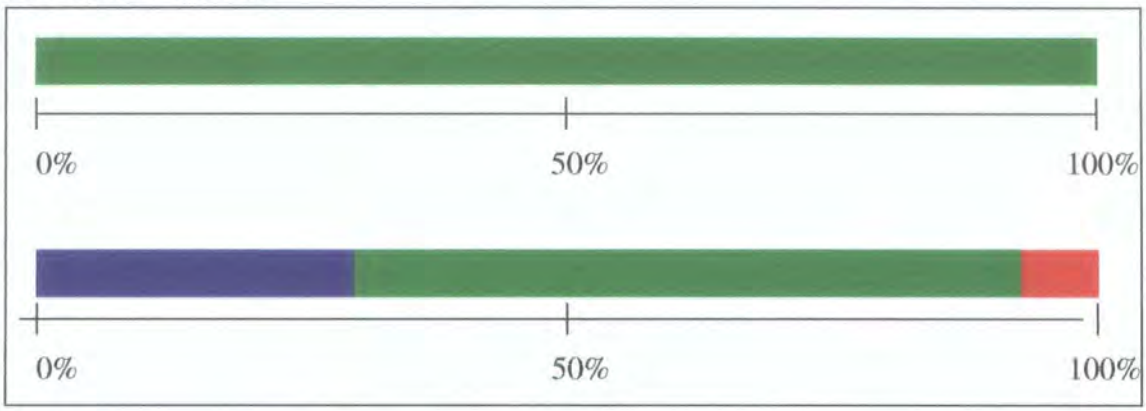


Figure 6-6. Visual representation of the cover of (top) a *pure* pixel (bottom) a mixed pixel

A-posteriori probabilities can be visualised in a similar way. After scaling the neural network outputs between zero and one and normalising them so that they sum to one, this method of representing pixel composition can be applied to each neural network output vector. If the reference data vector and the corresponding neural network output vector for a pixel are coloured appropriately and displayed side by side, the correspondence between the two can be visualised. If all the pixels in a data set are processed in this way, the correspondence between ground data and neural network outputs for all the pixels can be analysed. Images produced in this way will be referred to as 'Correspondence' images. The procedure for creating ground data/neural network *Correspondence* images is illustrated in figure 6-7.

Neural network experiments were run on the 'Portugal sixteen class', 'Portugal

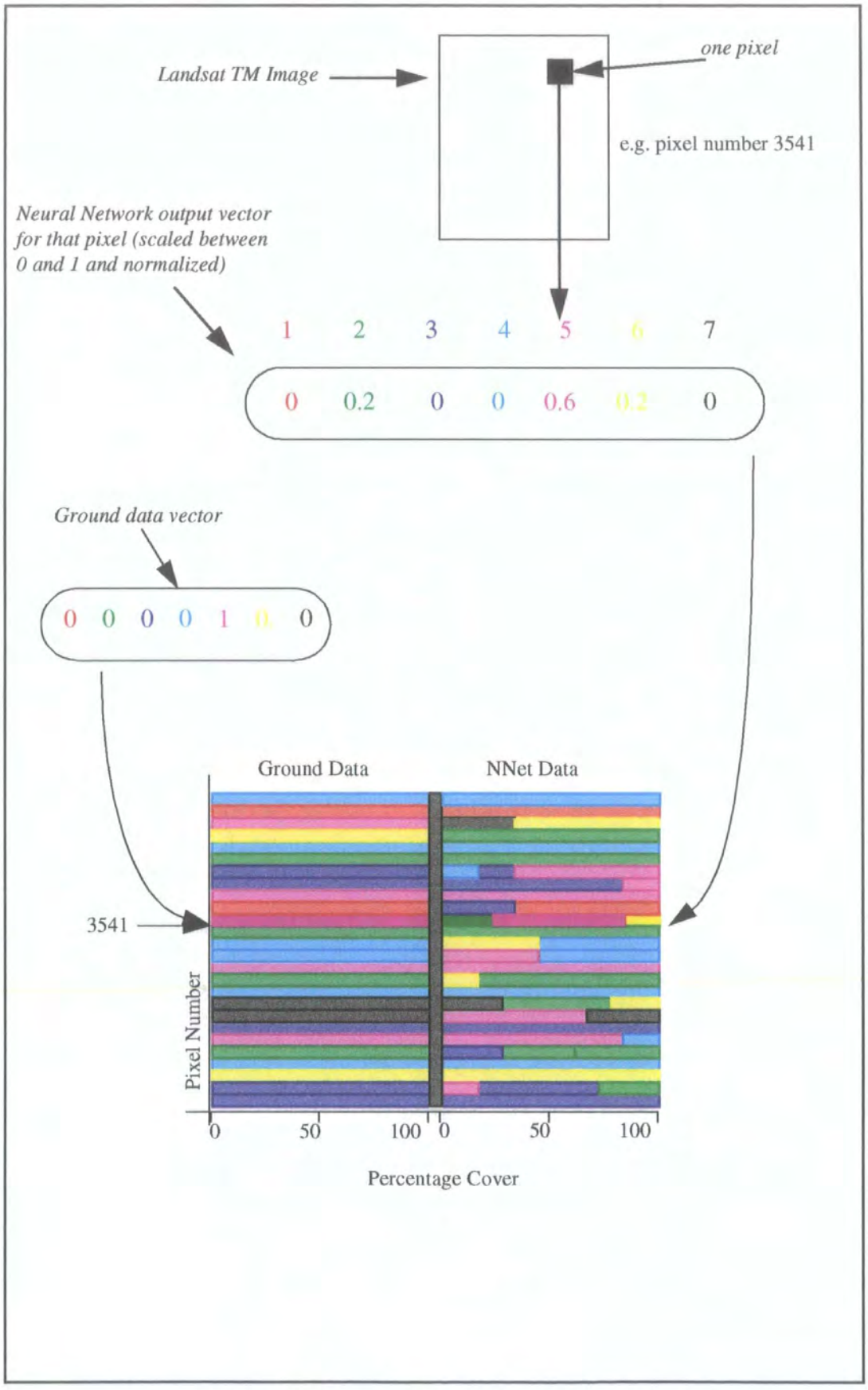


Figure 6-7. Procedure for creating ground data / neural network *Correspondence* images

fifteen class' and 'Portugal seven class' data sets. The 'Portugal sixteen class' data set was kept in a ratio of training to testing files of 2:1 since it had been provided in that ratio. The other two data sets were divided into ratios of 1:2. The composition of the 'Portugal sixteen class' data files was provided in chapter V. The composition of the 'Portugal fifteen class' and 'Portugal seven class' data sets are provided in appendix B, section B.5 and section B.6.

Parameters	Portugal 16 Class	Portugal 15 Class	Portugal 7 Class
Number of iterations	600	600	600
Architecture - Input Node **Hidden Nodes** **Output Nodes**	6 28 16	6 28 15	6 13 7
Type of weight initialization	Constant seed value	Constant seed value	Constant seed value
Learning rate	0.1	0.1	0.1
Randomisation	Once	Once	Once
Division ratio	TR : TS = 2 : 1 = 8,047 : 4,458	TR : TS = 1 : 2 = 9,217 : 18,434	TR : TS = 1 : 2 = 3,951 : 7,901

Table 6-5. Parameter settings for the classification of the 'Portugal sixteen class', 'Portugal fifteen class' and 'Portugal seven class' data sets

6.3.2 Results and analysis

'Portugal sixteen class'

Figure 6-8 shows the *Correspondence* images created from the 'Portugal sixteen class' data set. As described in chapter 4, this data set is typical of a conventional *pure* classification problem data set. In other words, all pixels are considered to be covered 100% by one class only. This is shown in figure 6-8 by the fact that, according to the ground data, all pixels are represented by a line of only one colour. The pixels were grouped by actual class, that is, the class to which the pixel belongs according to the ground data. Since the reference data assumes each pixel to be *pure*, the results can only be interpreted in terms of misclassification and not in terms of mixed pixels.

From figure 6-8, it is possible to tell which classes are best classified and with what classes they are confused. For example, the different water classes, **Sea Water**, **Estuary** and **Fresh Water** and the **Vines**, **Shrubs** and **Weeds** classes are well classified, since for the majority of the pixels of each of these classes, the dominant neural network class is correct. The **Water** classes are surprisingly well separated since it was expected from their spectra that **Sea Water** and **Estuary** would be confused. On the other hand,

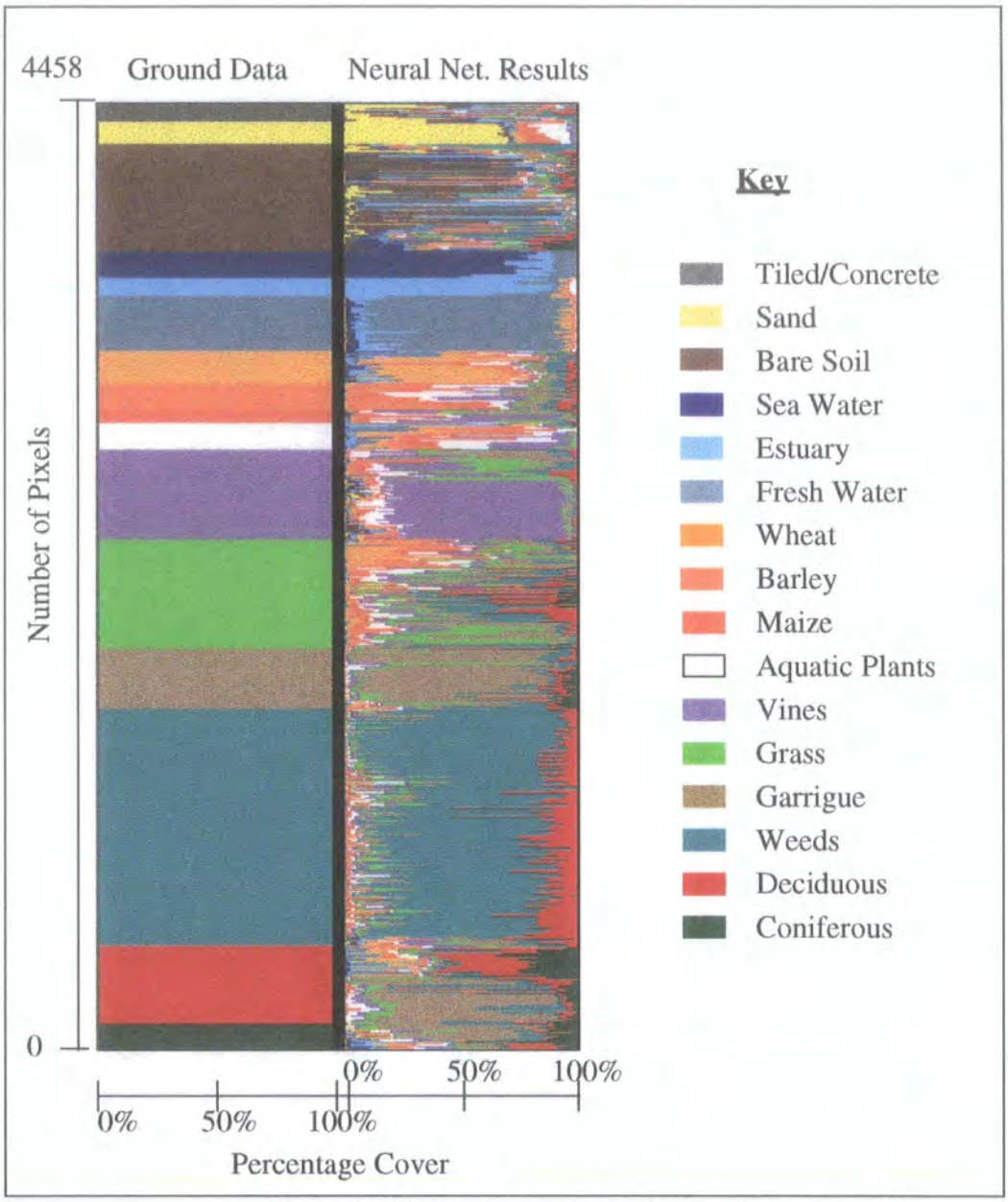


Figure 6-8. ***Correspondence*** **image for the 'Portugal sixteen class' testing file**

remaining classes are less well identified. Some such as **Sand** and **Wheat** have quite high output values in the correct class but not as high as the previously mentioned classes. The **Coniferous** forests and most of the **Deciduous** forest seem to be misclassified as **Shrubs**. This confirms the expectations set out in chapter IV. Classes such as **Bare Soil** and **Grass** seem to be confused with almost all the other classes.

The problem with this data set is that although it is known that the training areas were not completely *pure*, there is no information about mixing provided in the test set. Therefore, it is not possible to decide whether the neural network is identifying other classes present within a pixel or confusing spectrally similar categories. For example, the

Wheat class contains signals which suggest probabilities of assignment to **Bare Soil**, **Weeds** and **Grass**. It could be that within the **Wheat** training areas there were pixels of **Bare Soil**, **Weeds** and **Grass** or it could be that within **Wheat** pixels there were areas of **Bare Soil**, **Weeds** and **Grass** or it could be that the spectral signature of **Wheat** is similar to that of **Bare Soil**, **Weeds** and **Grass**. Without reference data against which to compare the results, no conclusions can be drawn.

'Portugal fifteen class'

The results for this data set are presented in figure 6-9. The image on the left shows the results on the testing file. To create the training and testing data sets, the original file containing all the data is randomised and then divided into two. Pixels in the testing file are therefore in random order. From this image, it is difficult to draw any conclusion because no pattern is discernible for the ground data and the large number of pixels involved (18,434) makes it extremely difficult to compare pixel by pixel. The testing file was therefore sorted and the resulting image is shown on the right of figure 6-9. The results are the same, only ordered differently.

The first point to note is that for most pixels, every node appears to register a minimum output value. For some pixels, the image shows that there is higher output in the correct class. For example, the pixels at the top of the graph which have a dominant **Coniferous** class, the **Water** pixels and the pixels containing **Vines** seem to register the highest response in the appropriate node. However, the network does not allocate classes with any degree of certainty and it is difficult to see the relationship between the ground data and the neural network outputs. The apparent confusion may be due to mixed pixels with different percentage cover being assigned the same target, which suggests the necessity for a different target representation. This issue is discussed in chapter VII.
The presence of a non-zero neural network value in all the nodes suggests the hypothesis that there might be a threshold associated with every output. In other words, the neural network perhaps produces a, possibly random, minimum signal at every output. The cause of this threshold might be the complexity of the data set, the variance associated with each class, target representation of the ground data, or the frequency of classes. It was therefore hypothesised that the nodes with the highest output strengths might represent classes which truly existed in the mixed pixel and that consequently, removing a threshold value from the network output strengths so as to favour those classes with the highest output strengths, would improve the identification of the sub-pixel components.

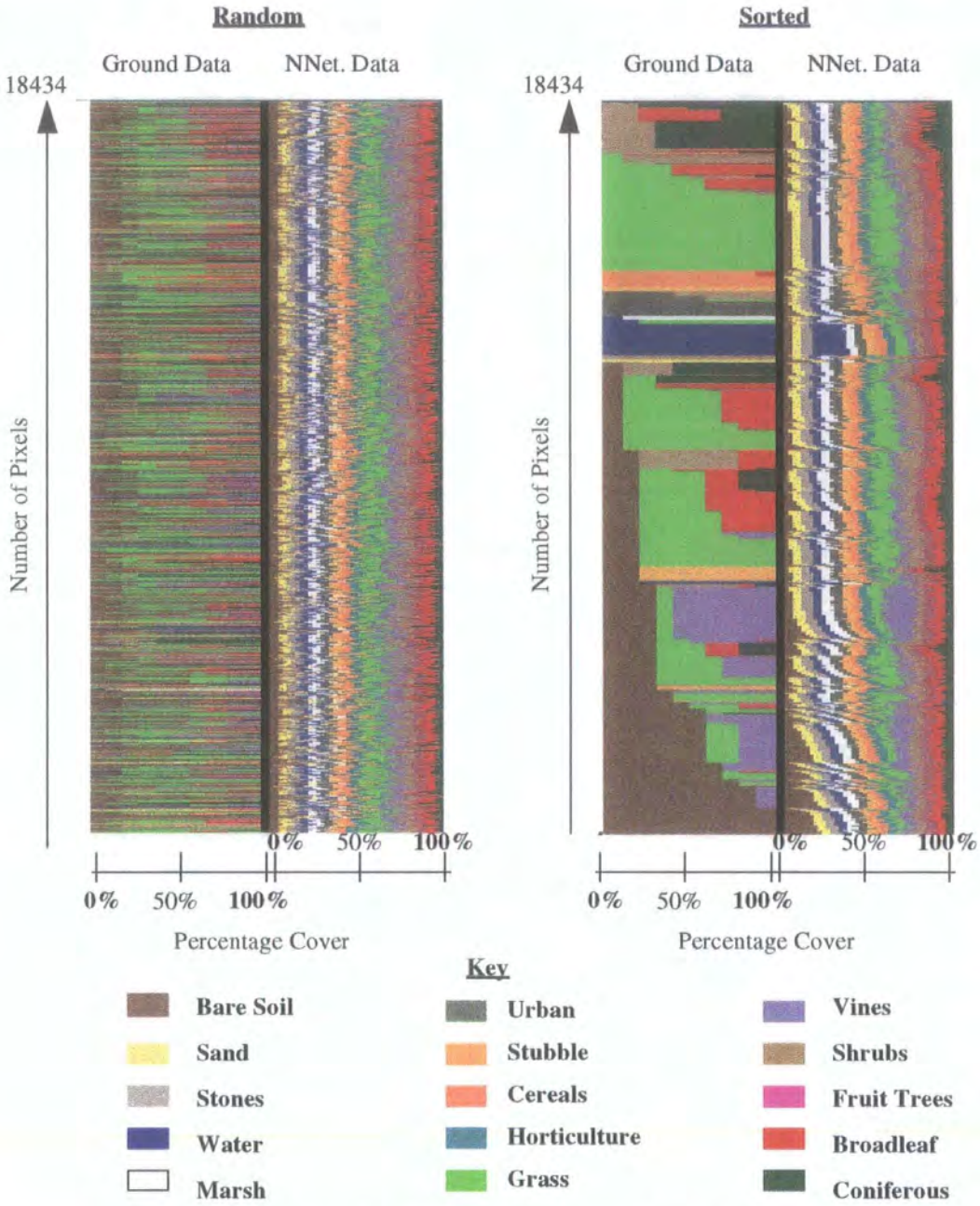


Figure 6-9. *Correspondence* images for the 'Portugal Fifteen Class' testing file. (left) random order; (right) sorted

'Portugal seven class'

The hypothesis was tested on the 'Portugal seven class' data set using a network with parameters as provided in table 6-5. Two methods of applying thresholds were tested. In the first method, a specified threshold amount is removed from all scaled neural network outputs. Neural network outputs which become negative are set to zero and the outputs are normalised. For the second method, neural network outputs which are less than or equal to a specified threshold are set to zero and then the outputs are normalised. An example of the two methods is given in figure 6-10.

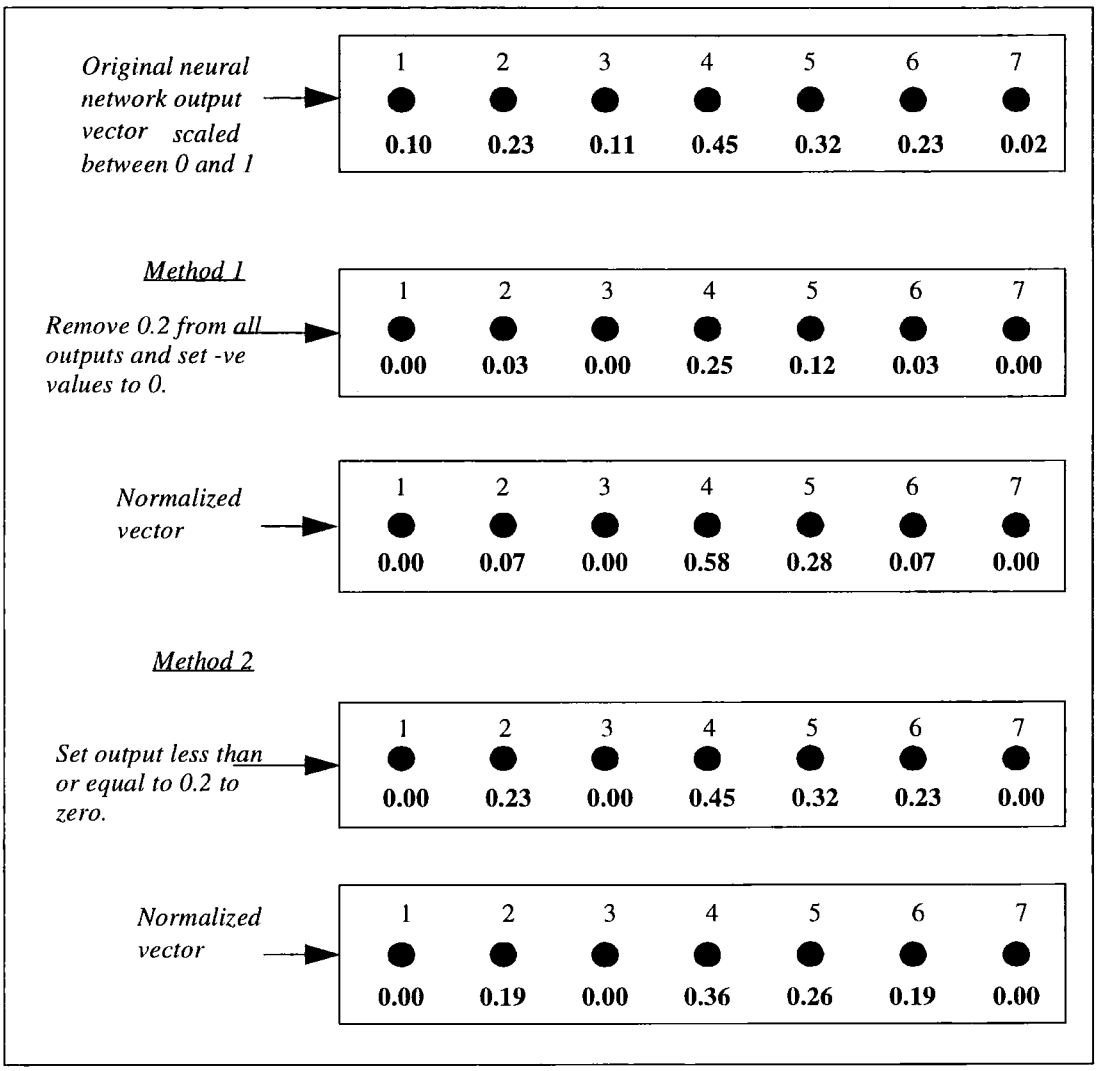


Figure 6-10. Illustration of the two methods of applying a threshold to the neural network outputs

The purpose of the first method was to investigate the effect of applying the same threshold to all the output nodes. The idea behind this method was that all output nodes for a pixel may be subject to the minimum signal. On the other hand, the second method was studied to determine the effect of applying a threshold only to the nodes with the smallest output strengths. The idea behind this method was that for each pixel, only the nodes whose class did not actually appear in the pixel were subject to this minimal signal.

Results are illustrated in figure 6-11. The strip on the far left shows the sorted image of the ground data and next to it the scaled and normalised neural network outputs. The strips to the right of this set show the images resulting from the neural network outputs after a threshold has been applied. The value of the threshold is specified at the top of each image; 'm1' refers to method 1. 'm2' refers to method 2.

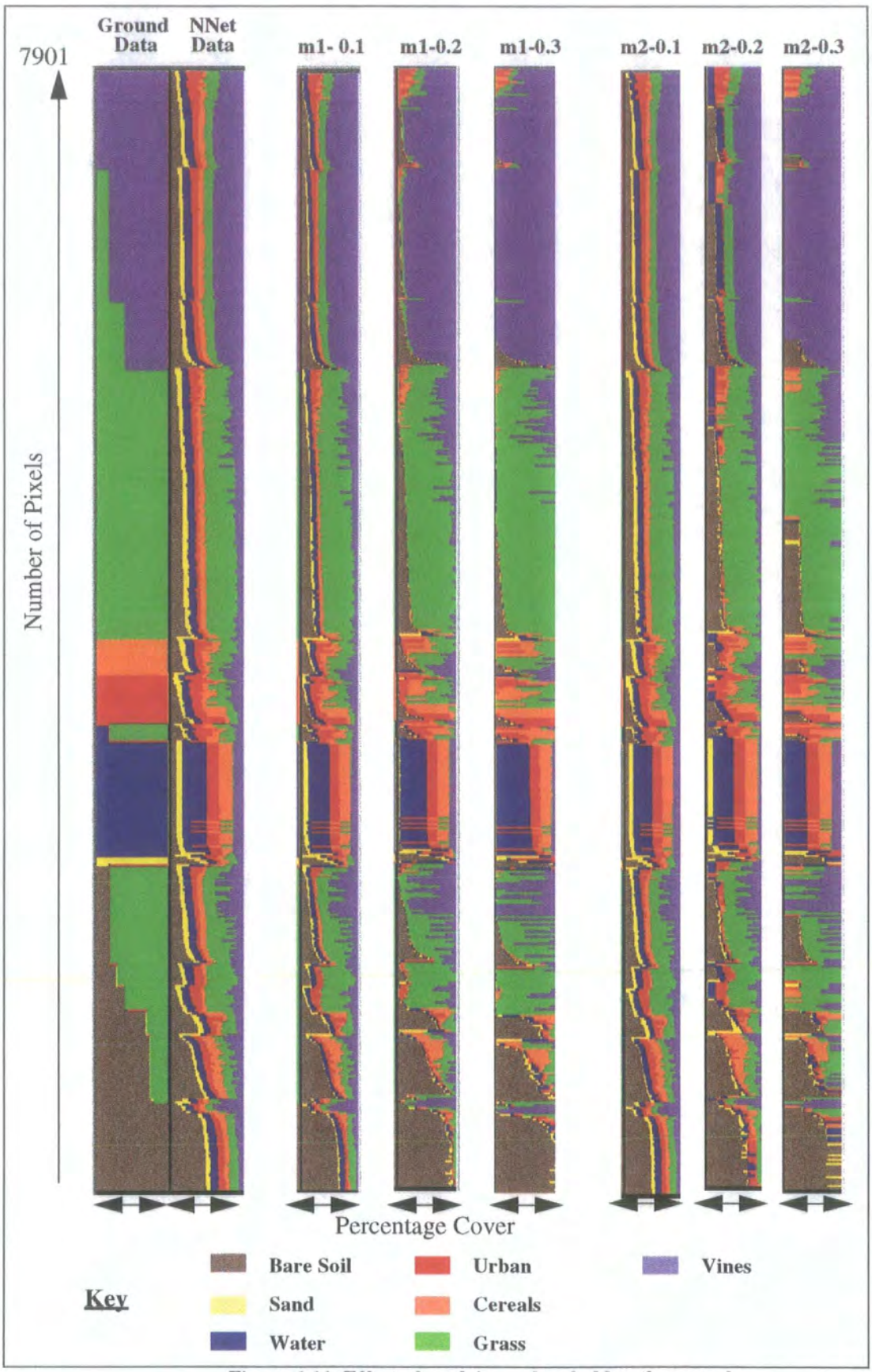


Figure 6-11. Effect of applying a threshold to the neural network outputs, using two methods, on the 'Portugal seven class' data set. Slight differences may occur from the resampling method of the image processing system used.

The images show that the neural network identifies the dominant class more clearly for this data set than for the previous data set. The output value for the dominant class is usually significantly larger than for the other classes, although it seems that most of the **Urban**, **Cereals** and **Sand** pixels may be misclassified. The second classes also seem to be better identified but increasing thresholds increase the extent of the dominant class and not always that of the correct second classes. Since the effect of the threshold is to remove low node values, this shows that the output strengths for the second classes are not necessarily always next, after the maximum value of the dominant class. In other words, the neural network does not always seem to correctly identify the second classes.

6.3.3 Summary

The *Correspondence* images are a useful visualisation tool which allows immediate comparison of neural network outputs with reference data. Sources of misclassification can be identified in the experiments using the 'Portugal sixteen class' data set. However, the large number of pixels and the complexity of the 'Portugal fifteen class' data set hinder analysis to determine whether second, third and so on, classes are correctly identified. The data seems to produce some confusion in the network results since all output nodes, for every pixel, register an output value. Analysis of the simpler 'Portugal seven class' data set by applying a threshold to the network outputs shows that the network output values are more related to the reference data than for the previous data set but the preliminary conclusion is that second classes are not always correctly identified. The confusion of the network may be caused by the type of target that was used. The use of quantitative analysis tools to provide a measure by which to compare results numerically are discussed in chapter VII and used to investigate different types of targets with which to train the network. The network may also require only *pure* pixels in the training stage, and mixed pixels may be confusing it. This issue is discussed in chapter VIII.

6.4 Neural Network Node Response Graphs

The *Correspondence* images allow the visual comparison between ground data and neural network outputs for individual pixels. The method developed here shows the overall distribution of responses at every node for a given class, directly in terms of the output strengths.

6.4.1 Methodology

The 'Portugal seven class' and then the 'Scotland eight class' data sets were used

for these experiments. The composition of the 'Portugal seven class' data set is provided in appendix B, section B.6. The 'Scotland eight class' data set was divided into a training file and a testing file in a ratio of 1:1 for the number of pixels in the training file to the number of pixels in the testing file. In addition, a set of training and testing files was created with a ratio of 1:1 containing only the first five classes which are all *pure* classes. Finally, a testing file containing only pixels from classes 6, 7 and 8 which are mixtures was also created. The composition of these files is reported in table 6-6.

		Scotland 8 Class		Scotland 5 Class		Scotland 3 Class
Id.	**Class Description**	**TR**	**TS**	**TR**	**TS**	**TS**
1	Closed Canopy Sitka Spruce	89	100	92	97	N/A
2	Closed Canopy Lodgepole Pine	107	102	96	113	N/A
3	Water	130	137	135	132	N/A
4	Background (Grass, Rocks)	309	328	315	322	N/A
5	Deciduous	211	207	222	196	N/A
6	Low Density Trees (mainly Sitka Spruce)	188	188	N/A	N/A	376
7	Medium Density Trees (mainly Sitka Spruce)	102	106	N/A	N/A	208
8	Closed Canopy Trees (Mixed)	229	196	N/A	N/A	425
Total Number of Pixels		1,365	1,364	860	860	1,009

Table 6-6. Composition of data files for the 'Scotland eight class', 'Scotland five class' and 'Scotland three class' data sets

Using the parameters listed in table 6-7, one network each for the 'Portugal seven class' and the derived 'Scotland five class' data sets were trained and tested. In addition, the network trained on the 'Scotland five class' was tested on the 'Scotland three class' data.

Parameters	**Portugal 7 class**	**Scotland 5 class**
Number of iterations	600	600
Architecture - Input Nodes **Hidden Nodes** **Output Nodes**	6 13 7	6 10 5
Type of weight initialization	Constant seed value	Constant seed value
Learning rate	0.1	0.1
Randomisation	Once	Once
Division ratio	1 : 2 = 3,951 : 7,901	1 : 1 = 860 : 860

Table 6-7. Parameter settings for the classification of the 'Portugal seven class' and 'Scotland five class' data sets

Node Response Graphs are created in the following way. The output strengths are plotted on the ***y-axis*** and pixels are grouped according to their dominant class, as provided by the reference data, on the ***x-axis***. There are usually, but not necessarily, as many expected classes as there are nodes. Each actual class is represented by a group of ***n*** nodes, where ***n*** is the number of *pure* classes the network has been trained for. For each pixel, the output value at every node is plotted.

6.4.2 Results and analysis

In the 'Portugal seven class' experiment, the network was trained to recognise seven *pure* classes so there are seven output nodes, and each of these classes was present in the testing file. Results of the classification of the 'Portugal seven class' data set are presented in figure 6-12, where each node is represented by a colour.

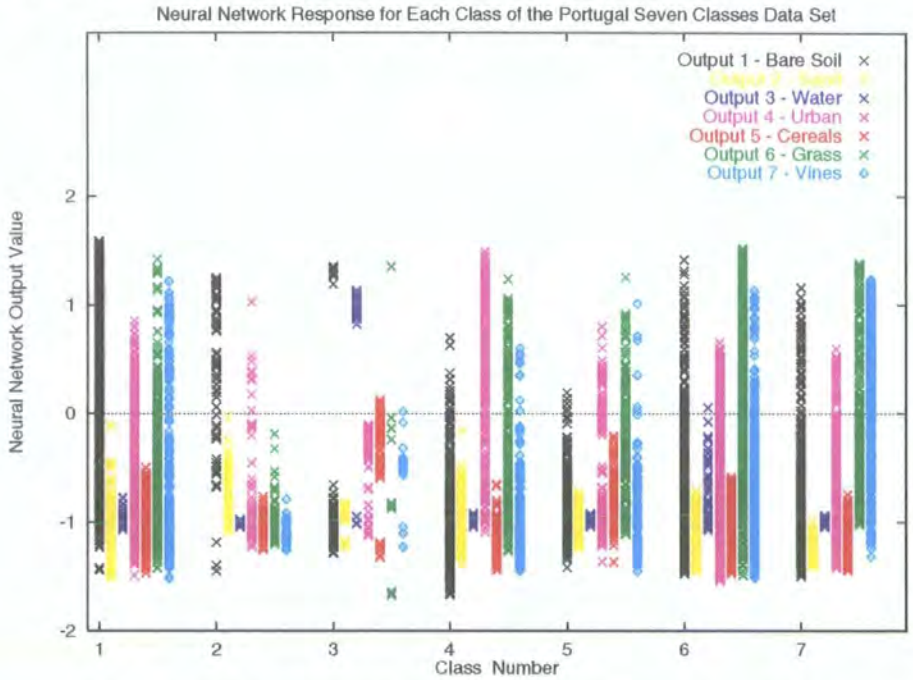


Figure 6-12. Neural network *Node Response Graph* for the 'Portugal seven class' data set

The first set of seven colours shows the output strengths at each node for all the pixels in the testing set which, according to the ground data, had a dominant class 1, **Bare Soil**. For example, the black node shows all the output strengths which were registered in node 1, corresponding to the node for **Bare Soil**; the yellow node shows all the output strengths which were registered in node 2 corresponding to **Sand** and so on for pixels whose dominant class is **Bare Soil**.

For *pure* pixels of **Bare Soil**, output nodes would be expected to register high values in the **Bare Soil** node and low values everywhere else. However, mixed pixels may register medium to high values in several nodes. In the data set **Bare Soil** is mixed with

Grass so some medium to high values would be expected in the **Grass** node for pixels whose dominant class was **Bare Soil**, and similarly, some medium to high values would be expected in the **Bare Soil** node for pixels whose dominant class was **Grass**. In fact, the graph shows that the node responses for pixels dominated by **Bare Soil** can be high for **Bare Soil** and **Grass** and also for **Urban** and **Vines**. In fact, **Bare Soil**, **Grass**, **Urban** and **Vines** classes are confused, or mixed, with one another for many pixels as can be seen from the graph which shows, for each of these actual classes, high node output values in all the corresponding nodes. The graph shows a degree of confusion between the spectral signatures of some of the classes.

The graphs shows that for the **Water** class, node 3 registers high output strengths whereas all the other nodes register low output strengths as would be expected in an accurate classification of pixels of this class. A few pixels register a high output value in the **Bare Soil** node which suggests that they may be misclassified. The cause of misclassification could be that the pixels were mistakenly labelled as **Water** in the ground data.

The node for the **Sand** class (class 2) never contains high values. Consequently it can be assumed that this class will be overwhelmingly misclassified. In fact, the **Sand** class registers high values for the node corresponding to **Bare Soil** which implies that **Sand** will be misclassified as that class. Going back to figure 6-11, which shows the same results displayed using *Correspondence* images, this is indeed the case as can be seen from the dominance of the **Bare Soil** class for the **Sand** pixels. A similar problem is encountered for the **Cereals** class which has no high values in its corresponding node. Pixels of this class register high **Grass** output values and some high **Urban** output values. Figure 6-11 shows this.

A similar analysis can be carried out on the results of the classification of the 'Scotland eight class' data set which was classified with an overall accuracy of approximately 84%. However, it is more instructive to look at the results of the classification of unknown classes 6, 7 and 8 with a network trained with classes 1 to 5. Classes 1 to 5 are **Closed Canopy Sitka Spruce**, **Closed Canopy Lodgepole Pine**, **Water**, **Background** and **Deciduous Trees** respectively. Classes 6, 7 and 8 are **Low Density Trees (mainly Sitka Spruce)**, **Medium Density Trees (mainly Sitka Spruce)** and **Closed Canopy Trees (Mixed)** respectively. These class names will be referred to as 1 - **Sitka Spruce**, 2 - **Lodgepole Pine**, 3 - **Water**, 4 - **Background**, 5 - **Deciduous**, 6 - **Low Density Trees**, 7 - **Medium Density Trees**, 8 - **Mixed Trees** in the discussion that follows. This experiment was run to determine the response of the neural network for

classes it had not been trained with, which were mixtures of classes it had been trained for. If the neural network output values reflect pixel composition, then:

- the outputs for class 6 - **Low Density Trees** pixels, a mixture of trees and grass, should be high for node 1 - **Sitka Spruce** and node 4 - **Background** and low for all the others.
- The outputs for class 7 - **Medium Density Trees** pixels should be high for node 1 - **Sitka Spruce** and node 4 - **Background** and low for all the others as for class 6. However, if classes 6 and 7 can be differentiated, then the output values in node 1 should be higher for class 7 than for class 6 and lower in node 4 for class 7 than for class 6.
- The output strengths for class 8 pixels should be high for node 1 - **Sitka Spruce** and node 2 - **Lodgepole Pine** and low for all the other outputs.

The neural network classified the five class data set with an overall final accuracy of 97%. It classified the eight class data set with an accuracy of 84%. Therefore, it would seem that the inclusion of classes which are not well defined reduces the accuracy of the network. Results of the classification of the 'Scotland five class' data set are displayed in figure 6-13.

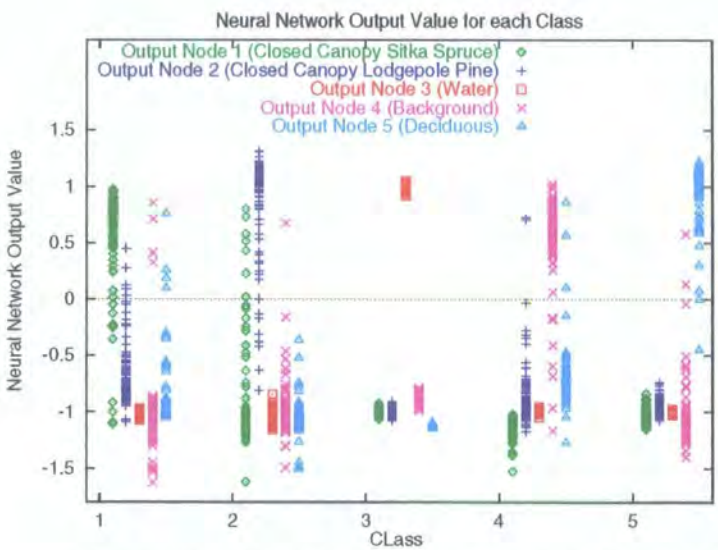


Figure 6-13. Neural network *Node Response Graph for the* 'Scotland five class' data set

Figure 6-13 shows that the **Water** class (3) has high output values in its corresponding output node and low values in all the others. Furthermore, the values are concentrated and there is little variance. The **Background** class (4) shows similar results although there are a few pixels with high neural network values in output nodes which do not correspond to **Background**. These could be pixels which are in fact covered by the

class which the neural network assigns to them which were perhaps misidentified in the reference data. The spread of values for each node is larger than that for the **Water** class. For the **Deciduous** class (5) similar results are seen: high output in the corresponding neural network output node, low output values in all the others. For the **Sitka Spruce** class (1), the corresponding output contains high values and other nodes low values as expected. However, the spread of values has increased and there are more outliers. In particular, the output values for the **Lodgepole Pine** class have a higher mean than the other non **Sitka Spruce** classes. These results are similar for the **Lodgepole Pine** class. Interestingly, the **Lodgepole Pine** class is less confused with **Deciduous** class than the **Sitka Spruce**. In summary, the classes appear to be quite well discriminated with little confusion and this is reflected in the high classification accuracy.

Results of the classification of the three class testing data set with the network trained on the five class data set are illustrated in figure 6-14. **Low Density Trees** is class 6, **Medium Density Trees** is class 7, and **Mixed Trees** is class 8.

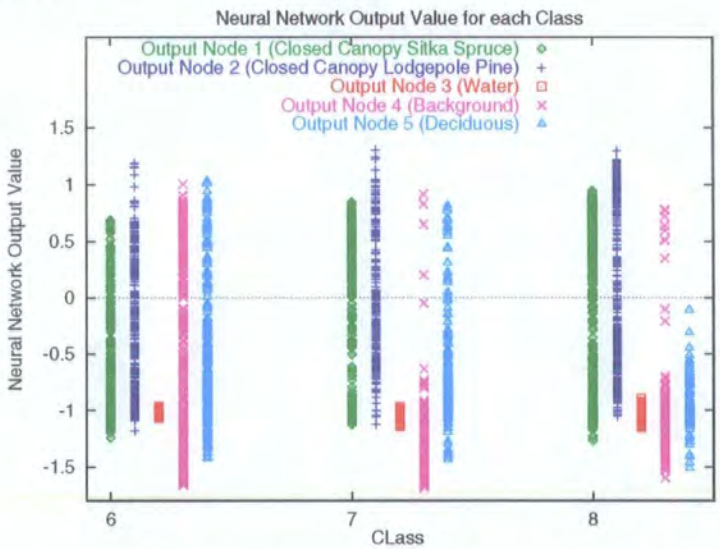


Figure 6-14. Neural network *Node Response Graph* for the 'Scotland three class' data set

The output values in node 3 (trained for **Water**) are very low for all three classes, showing that the network is definite that the pixels do not belong to this class. Class 8 - **Mixed Trees** registers high output values in node 1 - **Sitka Spruce** and node 2 - **Lodgepole Pine** and low outputs in all the others, as hypothesised. However, the spread of values is quite large, particularly for the coniferous tree nodes, which indicates confusion. In class 6 - **Low Density Trees**, the output values for the **Background** node are higher than for the other classes, again, as expected. On the other hand, the output values for this node are very low in class 7 -**Medium Density Trees**. This may mean that there is a threshold amount of second class component below which the network no longer

recognises the signal. Alternatively, it may be that in the training areas for **Low Density Trees** there were actually some *pure* **Background** pixels which are therefore classified as such by the neural network. The **Deciduous** forest node has some high values for classes 6 and 7. This implies that the background signal in **Low Density Trees** and **Medium Density Trees** affects the coniferous tree signal to a degree that they can be confused with deciduous trees. The **Mixed Trees** which are closed canopy had no high signals for the **Deciduous** node, which would support this hypothesis.

6.4.3 Summary

This method of visualising the output strengths from each node, for each class in a testing set, is particularly useful to examine the overall response of the network to the pixels. The results from this section have shown that the network does appear to respond to components in a mixture. A *Node Response Graph* for pixels in the 'Portugal seven class' data set showed that for well defined classes, output values were high only in the appropriate node. For classes that were confused, network output values were high in the incorrect node. Classes that included some mixed pixels, as well as possible spectral confusion, showed high responses in more than one node.

A *Node Response Graph* for the 'Scotland three class' testing data set, classified with a network trained only on *pure* classes, showed that the output values for the **Low Density Trees** class in the **Closed Canopy** and **Background** nodes were high and low elsewhere. Node values for the **Closed Canopy Mixed** class were high only for the **Closed Canopy** nodes and low elsewhere. This suggests that the network can extract component classes from mixed pixels, having only been trained with *pure* pixels. On the other hand, node values for the **Medium Density Trees** class were high in the **Closed Canopy** classes but low elsewhere. The expected high values in the **Background** node do not occur. There were also some high node values for the Deciduous node for **Low Density Trees** and **Medium Density Trees**. Therefore, it seems that the unmixing procedure is not straight forward.

There are three main limitations to the *Node Response Graph* method of anlaysis. The first is that it is not possible to determine the response for a particular pixel. In other words, the relationship between the output strengths of the different nodes for a single pixel is lost. This fact leads to the second limitation which is that if many of the pixels are confused between all the classes, the spread in the values at each node will be very large and no pattern will be obvious. Furthermore, it is not possible to tell the frequency of pixels at a particular point from the graph. For example, if fifty pixels whose dominant

classes is 5, produce an output strength of 1.2 for node 5, say, and only one pixel produces an output value of 0.3 for node 5, each of these values is only represented by one point on the graph.

6.5 Summary of Chapter VI

This chapter has addressed two issues: a study of the neural network outputs in the presence of mixed pixels and the development of visual analysis tools.

First, in section 6.1, statistics concerning correctly and incorrectly classified pixels within a *pure* data set, 'Portugal sixteen classes', showed that correctly classified pixels had higher maximum output values and lower next maximum output values than incorrectly classified pixels. Incorrectly classified pixels are likely to be mixed or contain a class whose spectral signature is similar to another class. This experiment shows that the network produces a different response for these pixels and that therefore it may be able to identify sub-pixel information.

Fraction images, discussed in section 6.2, similar to those created in linear un-mixing methodologies, show that the abundance of classes in certain areas as expressed by the network *a-posteriori* probabilities is as expected from the image. However, although they are more informative than a single *pure* classification image, proportion images only provide an overview of the distribution of classes. Degrees of mixing between classes are not obvious and they do not allow the direct comparison between ground data and results. Three-class colour images do not particularly alleviate the problems. For this reason, new visualisation techniques were developed.

The first technique, ground data / neural network *Correspondence* images described in section 6.3, provides a useful visualisation tool for comparing neural network outputs with reference data. Possible areas of misclassification were easily identifiable for the 'Portugal sixteen class' data set. However, the large number of pixels and the complexity of the data hindered analysis and seemed to produce some confusion in the network results for the 'Portugal fifteen class' data set. Results were improved with the use of the 'Portugal seven class' data set but remained unclear. The use of different target types to determine whether they may the cause of confusion are examined in chapter VII.

A different technique was used to study the response at each node for every pixel in a testing set: *Node Response Graphs*, discussed in section 6.4. The graph for the 'Portugal seven class' data set showed that well defined classes with no mixed pixels produced high output values at the correct node and low output values at the other nodes. Possibly poorly defined classes with no mixed pixels produced high output values at the wrong node and

low values elsewhere. Classes with mixed pixels and possible spectral confusion with other classes produced high output values in several nodes. Results from a network trained with a *pure* data set from the Scotland site and tested on untrained for mixed pixels were also shown in a *Node Response Graph*. The graphs showed that the neural network may be able to extract component signatures from mixed pixels in a simple mixture although the unmixing is not straight forward. Whether network training files should contain mixed pixels or only *pure* pixels is investigated in chapter VIII.

These results are somewhat different from those in the literature that suggest not only that the neural network can correctly identify mixture components but also their percentage cover of the pixel (Atkinson *et al.*, 1997; Foody *et al.*, 1997; Warner and Shank, 1997). However the experiments described in the literature tend to use very simple data sets with few classes and few pixels. The fact that conflicting results have been presented here may be a result of the complexity of the data sets. Part of the problem is also that the methods outlined here are qualitative. Quantitative methods which use some form of measure against which to compare results may provide a clearer answer. The next chapter, chapter VII presents quantitative analysis tools which are used to determine whether a more conclusive answer, as to whether a neural network could identify sub-pixel components, can be obtained.

Chapter VII

QUANTITATIVE ANALYSIS TOOLS AND TARGET TYPES

In the previous chapter, neural network outputs were analysed qualitatively using a combination of existing and newly developed visualisation tools. The methods described are useful for spatial analysis (fraction images), for a rapid visual analysis of the results which retains the relationship between classes within pixels (*Correspondence* images) and for an overview of the network response at every node as a function of the actual dominant class of pixels (*Node Response Graphs).* However, qualitative analysis alone is not sufficient. A numerical analysis is also necessary to provide actual values of classification accuracy. This chapter is concerned with quantitative techniques to evaluate the performance of the neural network for *soft* classification.

As Wang (1990a) states, measuring accuracies of fuzzy classification is the most challenging aspect of mixture classification. Traditional confusion matrices, the most common method of classification analysis, cannot deal with the added dimensionality brought about by mixed pixels. As discussed in chapter II, different techniques have been suggested. *Hardening* the results from a *soft* classification consists in only considering the maximum output value from the vector of *soft* outputs. In effect, this methods reverts to *pure* interpretations of classification (Foody, 1992; Wang, 1990b). Measures to compare performance of *soft* classifiers have been investigated including 'Entropy' (Maselli *et al.*, 1996) and the 'Closeness' between land cover and fuzzy membership values calculated for example by a Euclidean distance (Foody, 1996a). However, 'Entropy' assumes that the reference data is pure and that only the classifier results are fuzzy and indices such as the 'Closeness' index only provide an overall value, not a class by class and position by position breakdown.

Several authors have developed partially *soft* tools for the analysis of classification results. In a paper on ordinal classification of sub-pixel forest cover, Foody (1994) divides forest cover into four groups - large, intermediate, small, very small - and allocates pixels to one of these groups. He then calculates a confusion matrix on this basis. However,

pixels cannot belong to more than one group. Gopal and Woodcock (1994) are concerned with a slightly different problem from *soft* classification, since they assume that all pixels can be assigned to only one class but that there is uncertainty in that class' allocation which is expressed by an integer number representing a linguistic concept. For example, 'Absolutely right' is given 5, 'Understandable but wrong' is given 2. Nevertheless, the concepts are similar and the authors then calculate matrices which provide the number of matches between opinions and the size of the differences between allocations. In a later article, Woodcock *et al.* (1996) use a similar technique for calculating the accuracy of assignment of secondary classes. Van Deusen (1995) considers that classification produces measures of certainty of class allocation and makes a contingency table for the results including the second closest classes, that is those which have the second highest probability of allocation. Element **[n][m]** in the confusion matrix shows the number of times class [n] had the highest probability of allocation and [m] the second highest probability of allocation. Fisher and Pathirana (1990) use one matrix for each class to study the correlation between fuzzy membership values and percentage cover of a class. Moody *et al.* (1996) calculate the frequency of second classes classified as dominant and vice-versa.

However, none of the articles provide a compact and portable method for analysing *soft* classification results. This chapter describes two new types of confusion matrices which have some similarities with the techniques described above and which provide a quantitative analysis of the neural network's ability to identify the correct components of a mixture. Even though the matrices are used for neural network output results in the thesis, they can be used to analyse the results from any *soft* classifier which produces a vector of *soft* output values.

In the training stage, the neural network is provided with pixel data and a target to aim for. The network only knows that pixels are different if their targets are different; each target identifies the group to which pixels belong. Targets are therefore a vital part of the neural network classification process since they provide the basis on which classes will be differentiated. In the literature, the few articles which discuss neural network approaches to *soft* classification usually provide the network with either a *pure* target (Moody *et al.*, 1996), or with targets which contain values of the percentage cover for each class scaled between 0 and 1 (Foody *et al.*, 1997; Atkinson *et al.*, 1997; Warner and Shank, 1997). The aim of the experiments discussed in this chapter is to compare the performance of networks trained with different types of targets to represent the reference data. The matrices are used to quantify the effect of using the different types of targets.

Preliminary work relevant to this chapter is described in Bernard and Wilkinson (1996 and 1997).

7.1 Accuracy Measurement Techniques

7.1.1 The traditional confusion matrix

The most common type of classification accuracy reported in the literature is based on some form of confusion matrix, also called error matrix or contingency table. Confusion matrices are used to compare the actual class according to the reference data and the class assigned by the classifier, in this case the neural network, for each pixel. Figure 7-1 illustrates this type of matrix. The confusion matrix consists of an ***nxn*** array where ***n*** is the number of *pure* classes. The vertical dimension represents the reference data. The horizontal dimension represents network results. The order is sometimes reversed (Lillesand and Kiefer, 1994; Rosenfield and Fitzpatrick-Lins, 1986b).

Values in the matrix refer to numbers of pixels. Matrix positions are referred to as **[n][m]** where ***n*** is the reference class and ***m*** is the classifier class. Thus, if a pixel from class ***a*** according to the reference data is correctly classified as class ***a*** by the classifier, the counter in the position of the matrix **[n=a][m=a]** is increased by one. If a pixel of class ***a*** according to the reference data is incorrectly assigned to class ***b*** by the classifier, the counter at position **[n=a][m=b]** is increased by one. If a pixel of class ***b*** according to the reference data is incorrectly classified as class ***a*** by the classifier, the counter at position **[n=b][m=a]** is incremented by one. When all the pixels have been examined and the relevant counters updated, measures such as overall accuracy and commission and omission errors can be calculated.

The accuracies of classification for individual classes are calculated by dividing the value in the relevant position of the diagonal (for example, for class 2, position **[n=2][m=2]**) by the number of which belong to that class according to the ground data. The overall accuracy of classification is calculated by summing all the members of the diagonals, the correctly classified pixels, and dividing by the total number of pixels. High overall accuracy does not imply that all classes were well classified, individual class accuracies must also be examined. Commission error represents the number of pixels from other classes which were incorrectly classified as a particular class. The omission error represents the number of pixels of a class which were incorrectly classified as other

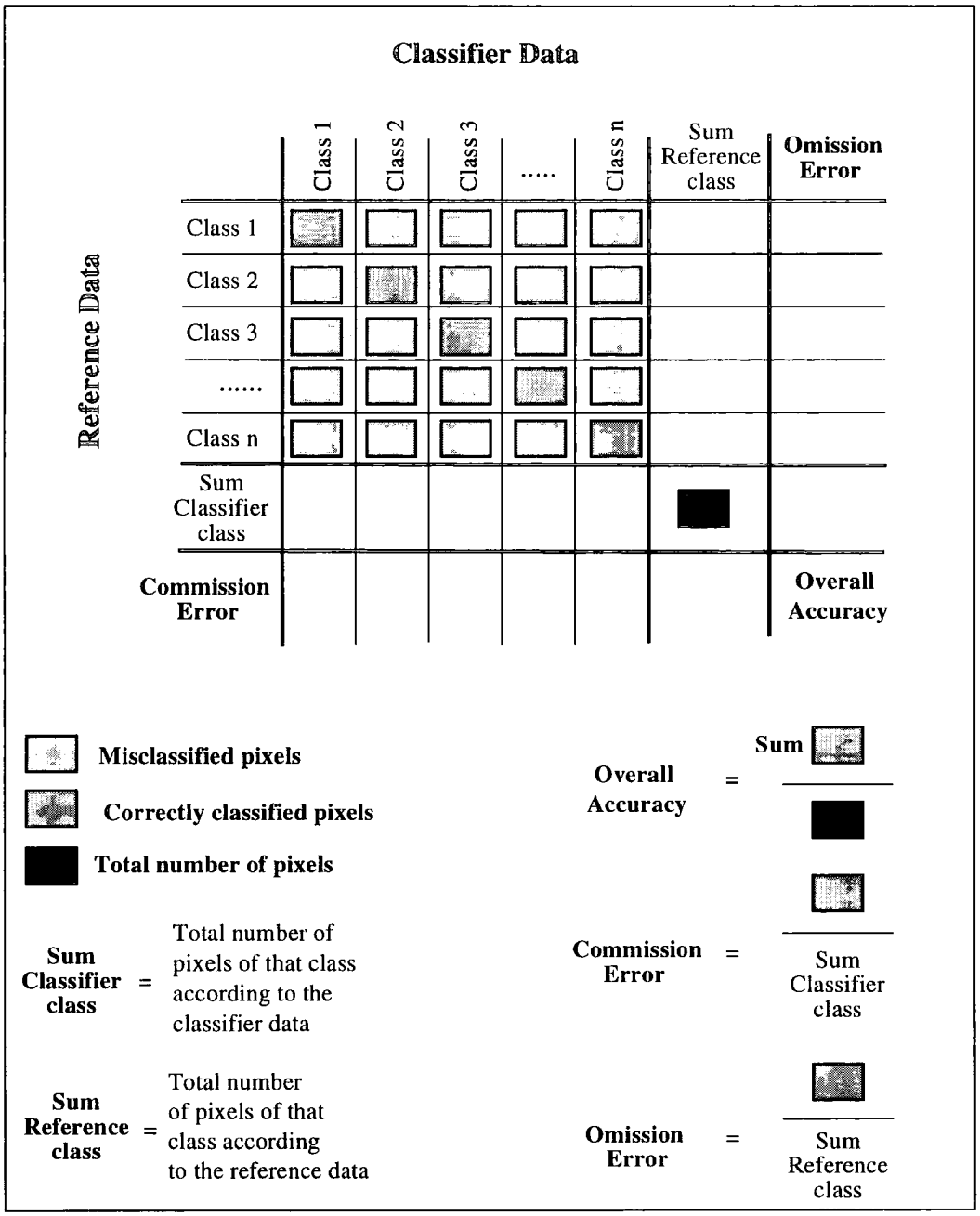


Figure 7-1. Traditional confusion matrix for *pure* classification analysis (adapted from Campbell, 1996)

classes. Commission and omission accuracies can be calculated from the matrix by dividing the correctly classified pixels of a class by the relevant row or column total.

A high overall accuracy means that training areas were homogeneous, training classes were separable, the testing set was representative of the training data and the classification strategy worked well with these training areas. It does not indicate how the classifier will perform for other sets of data from the same image (Lillesand and Kiefer, 1994). The assumption that is made is that the training and testing data are representative of the data from all the image and that therefore the confusion matrix represents results which would be obtained with any similar data set from the image.

7.1.2 The *Rank* matrix

The problem with traditional confusion matrices is that they can only deal with the dominant class of pixels. In mixed pixel classification, the identification of secondary or more classes is relevant. In this work, a new type of matrix was developed which will be referred to as a *Rank* matrix. It is complemented by a modified traditional confusion matrix.

The *Rank* matrix calculates the number of times positions, that is dominant class, secondary class and so on, are correctly identified; if they are not identified correctly, the *Rank* matrix shows the position to which a class was assigned compared to its actual position. The process to create the matrix is described below.

Figure 7-2 shows a hypothetical example of an output file for a group of mixed pixels from a data set described by five *pure* classes.

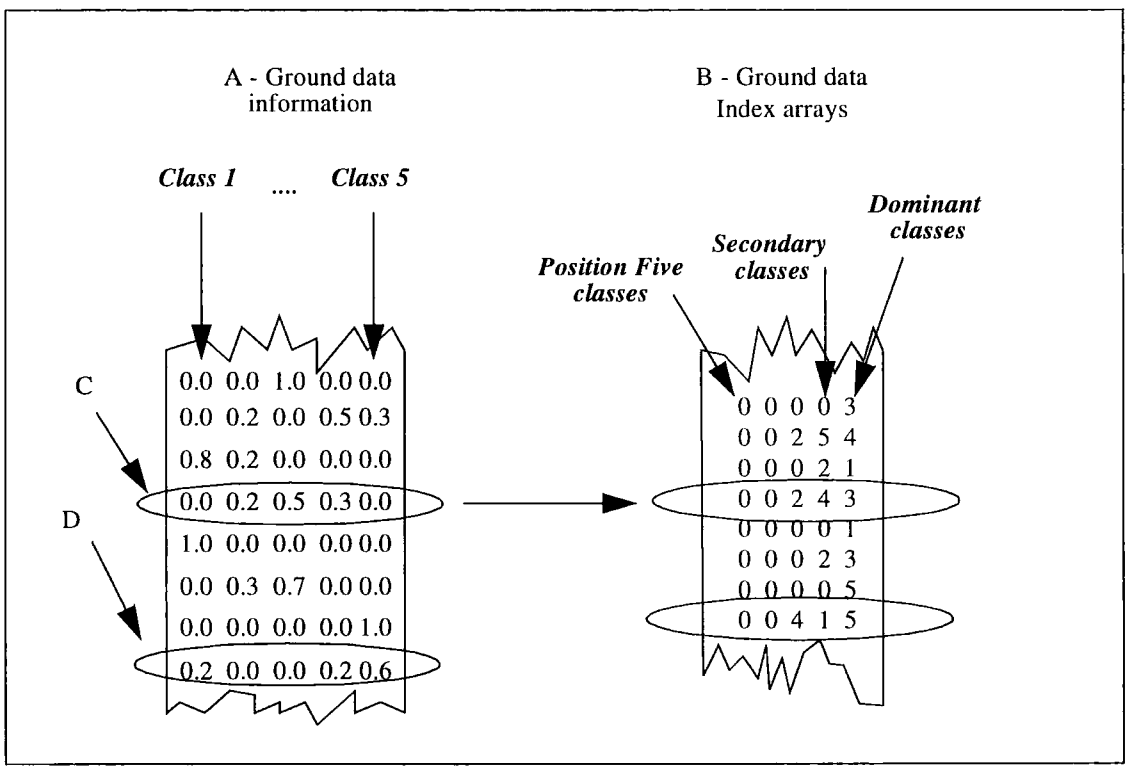


Figure 7-2. Procedure for creating ground data *Index* arrays

The ground information vectors for the pixels are shown on the left (**A**); percentage cover has been scaled between 0 and 1. From these, ground data 'Index' arrays as they shall be called, are created by sorting classes in increasing order of percentage cover according to the ground data (**B**). For example, the pixel indicated by '**C**' in figure 7-2 is mostly covered by class 3, followed by class 4, followed by class 2. These classes are ordered from least dominant to most dominant from left to right, that is 2, 4, 3, and placed in the *Index* array. A minimum percentage cover, usually 0%, below which the order of classes is considered irrelevant, is set and classes with that, or less than that, percentage are

ignored. In the example, classes 1 and 5 are not present in the pixel and therefore they are not recorded in the *Index* array. If two or more classes cover the same percentage of a pixel, for example classes 1 and 4 in pixel '**D**', the algorithm orders them from left to right from the last equal cover class to the first it came accross. Here, class 4 is placed as a third class of the pixel and class 1 as the secondary.

Figure 7-3 illustrates the similar procedure which is carried out with the neural network results.

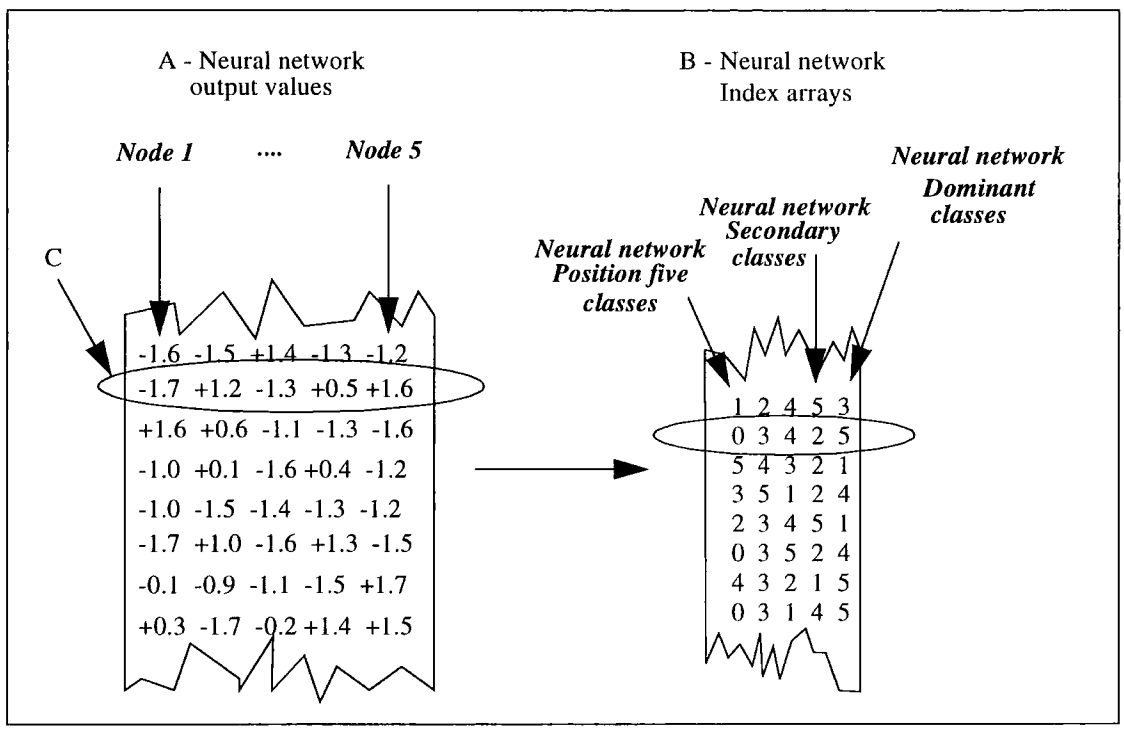


Figure 7-3. Procedure for creating neural network *Index* arrays

The neural network output vectors (**A**) are sorted in increasing order of neural network output values. A minimum value below which the order of classes is considered irrelevant, for example -1.7, is set. Classes are ranked in the correct order to form the neural network *Index* arrays (**B**). For example, according to the neural network results, the pixel indicated by '**C**' in figure 7-3, contains each of the following classes in increasing order of network output strengths: 3, 4, 2 and 5. The network output value for class 1 is equal to the threshold of -1.7 and therefore ignored by setting that position to 0.

Once the *Index* arrays have been created, they can be compared to determine whether the neural network identifies the components of the pixel in the correct order. Figure 7-4 illustrates the process. An array, called the 'Comparison' array records whether a position, dominant, secondary and so on, has been correctly classified or not. To create the *Comparison* arrays, the ground data *Index* arrays (**A**) and the neural network *Index* arrays (**B**) are compared for each pixel. When a class is correctly positioned by the

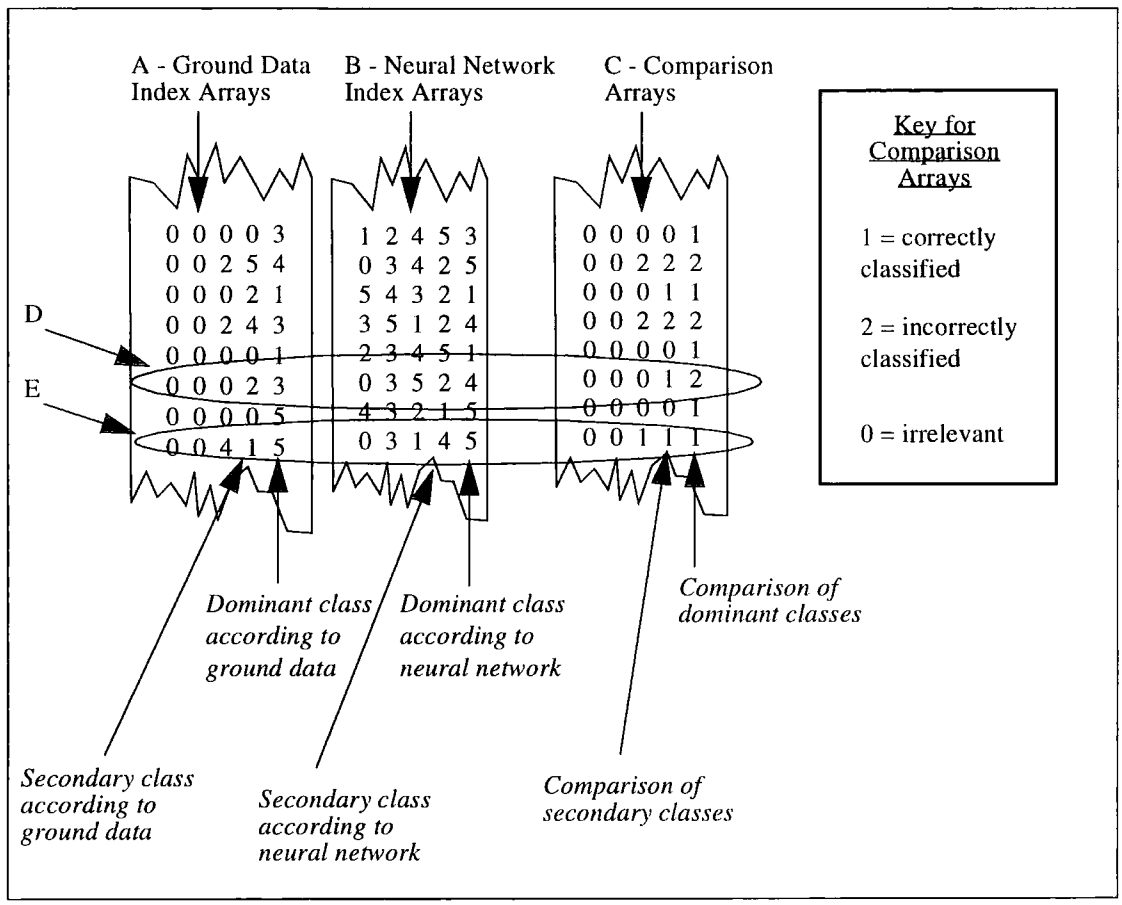


Figure 7-4. Procedure for creating the *Comparison* arrays

neural network, the element of the *Comparison* array (**C**) for that position is set to one. If the class is incorrectly positioned, the corresponding element of the array is set to two. If the position is irrelevant according to the ground data, the element of the array is set to zero. For example, for the pixel indicated by '**D**' in figure 7-4, the dominant class is not identified by the neural network, and consequently flagged with a two in the *Comparison* array, but the secondary class is correctly identified and consequently flagged with a one. According to the ground data, there are no other classes contained in the pixel so remaining elements of the *Comparison* array are assigned zero.

When two or more classes have the same percentage cover, this is taken into account. If one class is positioned as the other, the relevant element of the *Comparison* array is set to one as correctly classified. For example, in pixel '**E**' in figure 7-4, classes 1 and 4 both cover 40% of the pixel as shown in figure 7-2. The ground data *Index* array ranks class 1 as a secondary class and class 4 as a third class because of the sorting algorithm. The neural network assigns class 4 as the secondary class. The routine to create the *Comparison* array checks the percentage values of each class, and seeing that classes 1 and 4 have the same values, assigns that position as one, that is correct.

From the *Index* arrays, and with the help of the *Comparison* arrays, a *Rank* matrix is calculated. The vertical dimension of the matrix represents the positions within a mixture according to the reference data. The horizontal dimension represents the positions within a mixture according to the neural network. The *Rank* matrix has dimensions **[n][m]** where ***n*** is the maximum number of components in a pixel according to the ground data and ***m*** is the number of neural network output nodes and therefore *pure* classes in the data set. The elements of the matrix are calculated as follows.

For each pixel, if the dominant class according to the ground data is ***a*** and the dominant class according to the neural network is ***a,*** then the neural network has correctly identified the dominant class and the counter at position **[n=1][m=1]** is incremented by one. If the dominant class according to the ground data is ***a*** but the neural network has assigned class ***a*** to a third position say, then the counter at position **[n=1][m=3]** is incremented by one and so on.

Once the dominant class of the pixel has been analysed, the secondary position is studied using the same method. If for a pixel, the secondary class is ***a*** and the neural network has also identified ***a*** as the secondary class, then position **[n=2][m=2]** is increased by one. If the secondary class is ***a*** but the neural network has assigned ***a*** to a fourth position, then position **[n=2][m=4]** of the *Rank* matrix is increased by one. All mixture components which the ground data considers relevant, that is that are not equal to zero, are studied in this way. Classes with equal percentages which have been ranked one as the other are considered correct and the relevant counter is increased accordingly.

Figure 7-5 shows the *Rank* matrix formed from the reference data and neural network output results shown in figure 7-2, figure 7-3 and figure 7-4. The rightmost elements of the ground data *Index* arrays show the dominant classes. In this column, there are five classes, shown in red, which are correctly identified by the neural network. Consequently, in position [1][1] of the *Rank* matrix, which lists the number of correctly identified dominant classes, the counter is set to five.

In pixel two, the dominant class, class 4, is assigned to position three by the neural network. Thus, position [1][3] of the *Rank* matrix is filled with a one. In pixel four, the dominant class, class 3, is assigned to position five by the neural network. Thus, position [1][5] of the *Rank* matrix is filled with a one. Finally, in pixel six, the dominant class, class 3 is assigned to position four by the neural network. Position [1][4] of the *Rank* matrix is therefore also assigned a one. Misclassifications are shown in green.

Having calculated the dominant classes, the same method is applied to the secondary classes. There are two correctly identified classes, shown in red. In pixel eight,

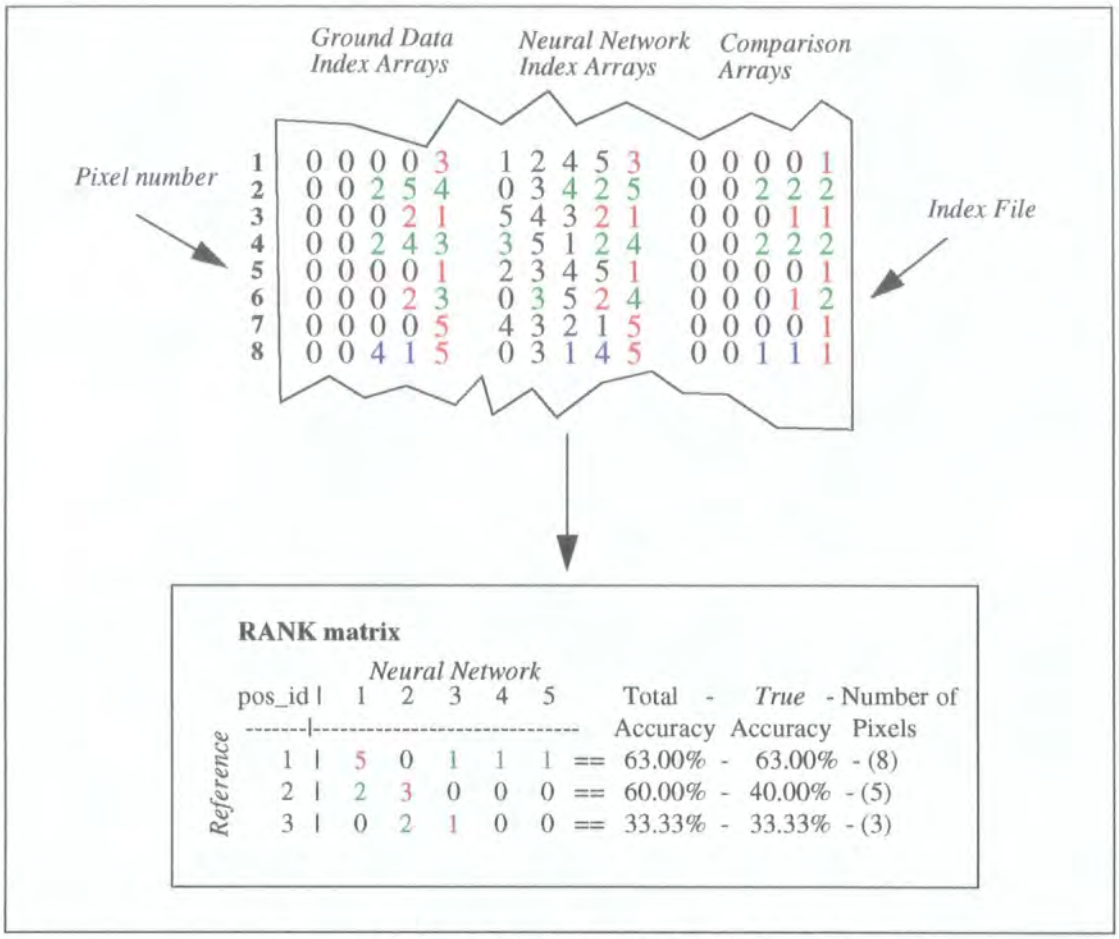


Figure 7-5. Illustration of the calculation of a *Rank* matrix for the hypothetical five class example

there are two secondary classes, classes 1 and 4, since they cover the same proportion of the pixel. The network assigns class 4 as a secondary class which is in fact correct, shown in blue. Pixel four and pixel two have their secondary class assigned to the dominant position by the neural network. The counter at position [2][1] is therefore incremented by two. The same method is applied to the third class and the matrix illustrated in figure 7-5 is complete. There are no further relevant positions according to the reference data.

In the matrix, row one represents the classification results for the dominant class, row two for the secondary class and so on. The overall percentage of correctly identified classes in that position is listed next to the appropriate row, under the column entitled 'Total Accuracy'. This is calculated by dividing the appropriate diagonal element by the total number of pixels containing that number of mixtures. For example, for three out of the five pixels that contain at least two categories, the secondary class was correctly identified. In other words, the secondary class of 60.00% of all the pixels that contain at least two classes was correctly identified by the neural network. The percentages are with respect to the number of mixture components in question, not with respect to the total

number of pixels. Since there may be less pixels with three components, for example, than pixels with two components, overall accuracy may be higher for the third position than the second position since the correctly positioned classes are divided by a smaller number.

It is possible that a secondary class of a pixel, say, is correctly identified although the dominant class of the pixel was incorrectly identified. For this reason, two total accuracies are reported. The first is the number of correctly identified classes for the relevant position. The second total accuracy, entitled '*True* Accuracy', is the number of correctly identified classes for the position where the previous position was also correctly identified. For example, the *true* accuracy for the secondary classes is 40.00% whereas the percentage of correctly positioned secondary classes regardless of the position of the first class was 60%.

Finally, next to these accuracies, the number of pixels with at least that number of mixture components is reported. In other words, the number of pixels containing at least one class, that is all the pixels, is listed for the first row; the number of pixels with at least two mixture components is reported in the second row and so on. In the example, there are eight pixels, five of which are covered by at least two classes and three of which have three components.

In summary, the *Rank* matrix provides the following information:

- for each position, dominant, secondary and so on, the number of pixels where the class was correctly identified, regardless of the class,
- for each position, the percentage of correctly identified classes; for positions beyond the dominant class, the percentage of pixels where the classes in the relevant position and all positions prior to it were correctly identified,
- for each position, the number of pixels containing at least that number of mixture components,
- in addition, the matrix allows the user to determine for how many pixels, classes in a position were incorrectly identified as belonging to another position and to which position.

7.1.3 The *Modified* misclassification matrix

The *Rank* matrix provides no information on the distribution of misclassifications between classes since only positions are compared, irrespective of the actual class. For example, although the *Rank* matrix illustrated in figure 7-5 shows that the dominant class of six pixels out of eight was correctly identified, it does not show what those classes

were. A *Modified* confusion matrix, figure 7-6, is used to analyse the performance of the network for each class, irrespective of the actual position of the class within pixels. The *Modified* confusion matrix is of dimensions ***nxn*** where ***n*** is the number of *pure* classes in the data set. The vertical dimension represents reference data. The horizontal dimension represents classifier data. Values in the matrix refer to numbers of pixels. The matrix elements are updated in the following way.

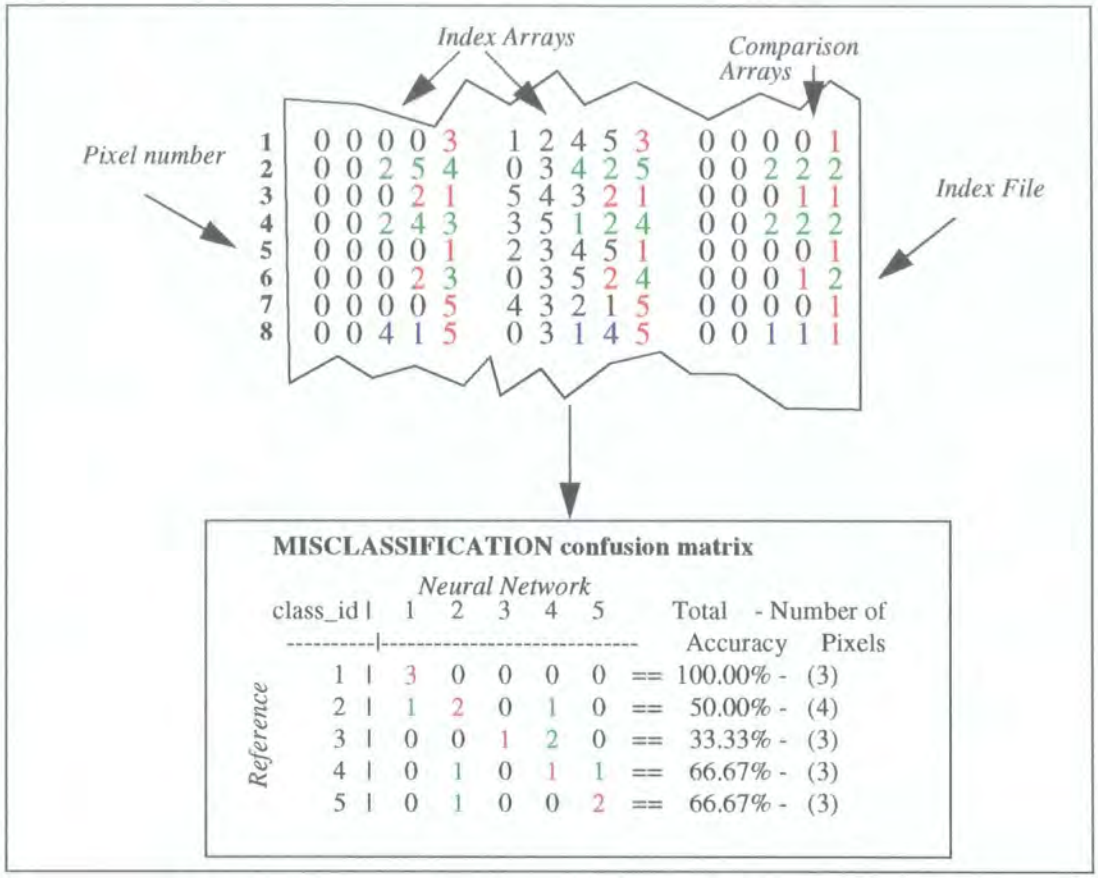


Figure 7-6. Modified traditional confusion matrix for the hypothetical five class example shown in the previous figures

If for a mixed pixel, the dominant class ***a*** according to the reference data is correctly classified as class ***a*** by the classifier, the counter at position **[n=a][m=a]** is incremented by one. If the dominant class is classified as anything else, the relevant position is updated in a similar way to traditional confusion matrices. However, once the dominant class has been analysed, the next component of the mixture is also considered. The class allocation for the secondary class is analysed as if it were the dominant class and matrix positions are updated accordingly. Therefore, regardless of the position of a class, its correct or incorrect identification are recorded in the misclassification matrix.

Figure 7-6 illustrates the *Modified* misclassification confusion matrix for the examples shown previously. Class 1 was correctly identified by the neural network twice. In addition, in pixel 8, classes 4 and 1 cover the same proportions and the classification is considered correct whether the network orders the classes as 1 followed by 4, or whether

it orders them as 4 followed by 1. Therefore, the total number of correctly identified class 1's is three. Class 1 is not allocated as anything else, nor are any other classes identified as class 1. There are two correctly identified class 2's; one class 2 identified as class 5, and one class 2 identified as class 1; and so on.

Overall accuracies for each class are calculated by dividing the number of correct allocations, identified in the relevant diagonal element, by the number of pixels containing the class. The column headed 'Number of Pixels' adjacent to each row indicates the total number of pixels containing the class of the row. Thus row one indicates the total number of pixels which according to the ground data contain class one. Similarly, row two indicates the number of pixels containing class two and so on. This type of confusion matrix allows the user to determine for how many pixels containing a class the network assigned the wrong component class, and what that class was.

The *Rank* matrix provides a breakdown of the position misclassifications; in other words, how the network orders classes compared to how classes are actually ordered according to the reference data. The *Modified* misclassification matrix, on the other hand, provides a breakdown of the class misclassifications; in other words, which classes the network confuses. The two matrices are best used in conjunction as the performance of a *soft* classifier is expected to be a function of class separability and class proportions within pixels. If two classes are very similar spectrally they are likely to be confused at the classification stage. However, even though two classes may be different spectrally, if they each cover half a pixel, the resulting signature is also expected to confuse the network.

7.1.4 Limitations

There are limitations to these methods of quantitative analysis. The *Rank* matrix does not take into consideration the percentage cover of classes other than for ranking purposes. For example, figure 7-7 illustrates three pixels with their relative proportions of classes according to the ground truth, and possible neural network outcomes. In the first two columns of neural network results all the pixel interpretations would be considered correct even though the percentage covers are very different; only relevant positions according to the ground data are considered when creating index arrays. On the other hand, the pixel interpretations which are labelled incorrect have similar percentage cover to the ground truth. The *Rank* matrix seems to exaggerate the importance of small mixture components for both the ground data and the neural network output values.

In addition, a class is considered dominant if it has the highest percentage cover even though it may not actually be covering most of the pixel. For example, a mixture of

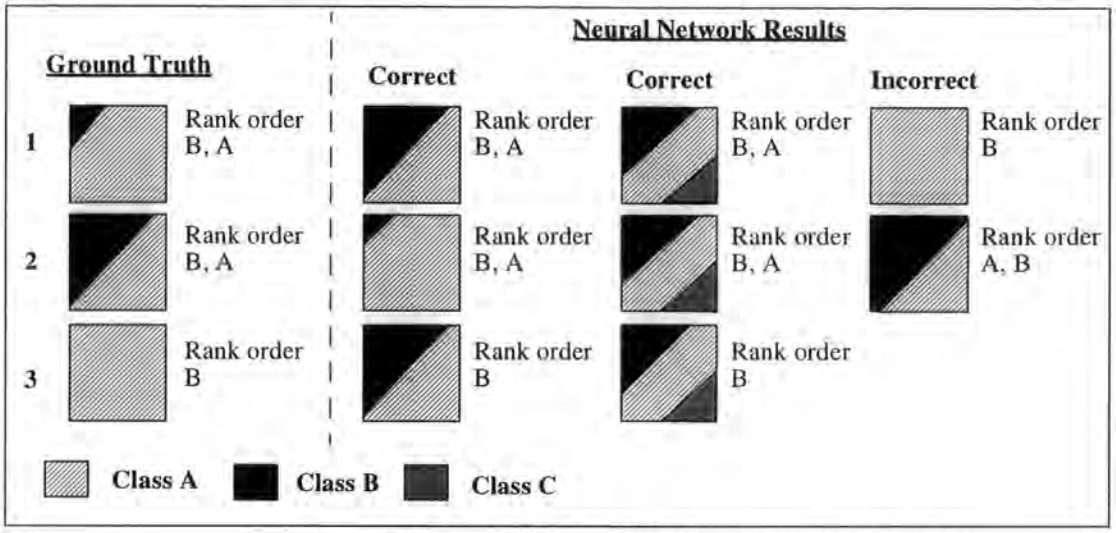


Figure 7-7. Example hypothetical ground truth and possible outcomes for pixels

[40% class 02 + 30% class 04 + 30% class 05] is considered by the sorting algorithm to have class 02 as the dominant class.

The *Modified* matrix can give accuracies which appear misleadingly low since the overall percentage accuracy is calculated over all positions, regardless of whether the 1st or the 4th for example. Therefore, if class 1 occurs in five pixels as the dominant class and fifteen pixels as the third class and the classifier identifies all the pixels with class 1 as a dominant class but not the others, the accuracy of classification will be 25%. If class 1 occurs in five pixels as the dominant class and in no other, and the classifier identifies all the pixels with class 1 as a dominant class, the accuracy of classification for class 1 will be 100%.

In summary, the accuracies which are reported by the *Rank* and *Modified* misclassification matrices may sometimes be misleading. Nevertheless, these two matrices provide a means to evaluate the performance of the network for *soft* classification. They provide information which traditional confusion matrices based on *hardened* results cannot provide about classes other than the dominant class.

7.2 Effect of Target Types on *Soft* Classification Accuracies

As discussed in chapter VI, the neural networks were run with traditional *pure* targets which treat mixed pixels as if they were covered 100% by the dominant class. For example, the targets for a pixel with a composition of [30% class 1 + 50% class 6 + 20% class 10] or [10% class 2 + 90% class 6] or [40% class 6 + 30% class 2 + 30% class 7] are assigned a value of +1 for class 6 and -1 for all other classes. This may be a source of confusion for the network since pixels with different class compositions are assigned the same target and some pixels are assigned to a class even though that class may not cover the largest area of the pixel. The fact that in the experiments in chapter VI, neural network output nodes all registered a small value whatever the expected pixel composition is thought to be in part due to the target type.

Having developed tools for the quantitative analysis of *soft* classification results, they can be applied to the problem of determining whether different target types have an effect on classification accuracy and what that effect may be. This section provides a systematic analysis of six different types of targets given the names: '*Pure*', '*Scaled(0,1)*', '*Scaled (-1,1)*, '*Bin (6)*', '*Bin (4)*' and '*Occurrence*'. The effect of the target types is evaluated by comparing the performance of networks run with constant parameters and data sets but different target representation of the ground information. The performance is measured using the *Rank* and *Modified* matrices.

7.2.1 Methodology

The *Pure* target is typically used in *pure* classification and is that employed up to now in the experiments described in chapters V and VI. Training data are partitioned into ***n*** classes where ***n*** is the number of *pure* classes. Regardless of the composition of a pixel, a dominant class is identified and assigned +1 in the target vector. The other classes are assigned -1 even if they also occur in the pixel. When there are two or more classes with the same percentage cover, the last class in the ground data array from left to right with that percentage cover is considered to be the dominant class. This type of target does not provide the neural network with any mixture information and is expected to confuse the network because classes that have similar proportions may be assigned different targets, different class compositions may be assigned the same target and some pixels are assigned to a class even though that class may not cover the largest area of the pixel. Dominant classes are expected to be poorly identified since mixed pixels increase the variability of spectral signatures within classes. Other class components within pixels are only expected to be correctly identified if their signal is strong enough to activate the correct second,

third or so on node. Furthermore, if a class only ever occurs as a secondary class or less, it will have no target associated with it.

In order to provide the network with mixture information, it must be incorporated in the target. Two types of *Scaled* targets were tested. The *Scaled* targets take ground cover information and scale the percentage cover between minimum and maximum values. For the first target, *Scaled (0,1)*, the percentage cover of each class is scaled between zero and one. The target vector consists of zeroes when classes are not present and scaled values of cover when classes are present. The second type of *Scaled* target, *Scaled (-1,1)*, consists of percentage cover scaled between -1 and 1. Classes which are not present in a pixel are assigned -1 and classes which are present in the pixel are assigned the scaled value between -1 and 1 of their percentage cover.

These two types of targets provide accurate mixture information to the network, pixels are partitioned into as many different groups as there are different mixtures, but require very accurate ground data and the neural network must produce very precise information. Furthermore, the first type of *Scaled* target, *Scaled (0,1)*, has a range which only lies in the positive half of the activation function. As discussed in chapter III, section 3.3.1, the activation function for the network used in this thesis is a sigmoid *tanh* function with a range between -1 and +1, scaled to improve performance so that neural network outputs lie between -1.7 and 1.7. The *Scaled (0,1)* therefore only utilises half the activation function and is expected to produce lower classification accuracies than the second type of target, *Scaled (-1,1)*. Although this type of target is that used in the literature, the authors use activation functions which lie between zero and one (Foody, 1996b; Atkinson *et al.*, 1997).

The third type of target was developed to reduce the need for very accurate ground data and precise neural network output values. Instead of precise percentage cover information, the *Bin* targets divide ground cover information into discrete ranges of percentages. For example 25% to 50% cover may be represented by only one value. This type of target is expected to be more flexible and less sensitive to inaccurate ground data information as it allows the neural network to produce a range of output values to represent the same percentage cover range. However, problems may arise when percentage cover values lie around the border between two ranges which although very similar in reality will be represented by two different *bins*. For example, if percentage cover between 50% and 75% is represented by **x** and percentage cover between 76% and 100% is represented by **y**, proportions of 74% and 77% will be assigned to the two different bins even though they are similar.

The *Bin* targets are created as follows. A number of *bins* is specified then the value of each *bin* is decided. For example, *bin* 1 = -1; *bin* 2 = +0.2 and so on. Then the range of cover proportions which fall within each bin is defined; for example, if cover is 0%, class belongs to *bin* 1; covers 1% to 25% belong to *bin* 2 and so on. When all the *bin* values and ranges have been defined, a target can be created for each pixel which describes the classes and their proportions which the pixel contains. Two *Bin* targets were tested whose characteristics are described in table 7-1. The first, referred to as *Bin (6)*, consists of six bins; the second, referred to as *Bin (4)*, contains four bins. The values of the bins were chosen to be symmetric about zero, so as to reflect the numerical range of the activation function.

Bin Number	Bin Values for Target Bin (6)	Bin Ranges for Target Bin (6)	Bin Values for Target Bin (4)	Bin Ranges for Target Bin (4)
1	-1.0	0%	-1	0-25%
2	-0.6	1-25%	-0.3	26-50%
3	-0.2	26-50%	+0.3	51-75%
4	+0.2	51-75%	+1	76-100%
5	+0.6	76-99%		
6	+1.0	100%		

Table 7-1. Description of *Bin(6)* and *Bin(4)* Targets

Finally, the *Occurrence* target was created to determine whether the neural network responded to the magnitude of its training target values or simply to the presence of a value not equal to -1 to indicate mixtures. The *Occurrence* target is created by setting all classes which occur in the pixel, regardless of percentage cover, to a value of 1 and all the others to a value of -1. This type of target is not expected to perform well since a mixture of [10% class 02 + 90% class 08] for example, will have the same target as a mixture of [90% class 02 + 10% class 08]. In some ways, this target is the opposite of the *Pure* target. The *Occurrence* target type assigns the same targets to pixels with different compositions whereas the *Pure* target assigns different targets to pixels with similar compositions.

An example of a mixture composition for a seven class problem and respective target representations for each target type are provided in table 7-2. Neural networks with parameters settings as in table 7-3 were trained with each of the target types on the 'Portugal fifteen class' data set. This data set was chosen because it includes details of the composition of mixed pixels and is typical of a land cover classification problem.

Target Type	Target Value
Actual Pixel Composition	*30% class 1 + 60% class 4 +10% class 6*
Pure	-1.0 -1.0 -1.0 +1.0 -1.0 -1.0 -1.0
Scaled (-1,+1)	-0.4 -1.0 -1.0 0.2 -1.0 -0.8 -1.0
Scaled (0, +1)	+0.3 0.0 0.0 +0.6 0.0 +0.1 0.0
Bin (6)	-0.2 -1.0 -1.0 +0.2 -1.0 -0.6 -1.0
Bin (4)	-0.3 -1.0 -1.0 +0.3 -1.0 -1.0 -1.0
Occurrence	+1.0 -1.0 -1.0 +1.0 -1.0 +1.0 -1.0

Table 7-2. Example of target representation of ground information by each target type

Parameters	Portugal 15 class
Number of iterations	600
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 28 15
Type of weight initialisation	Constant seed value
Learning rate	0.1
Randomisation	Once
Division ratio	TR : TS = 2 : 1 = 9,701 : 18,434

Table 7-3. Parameters for the neural networks using different target types

7.2.2 Results and analysis: 'Portugal fifteen class' data set

The *Rank* and *Modified* confusion matrices for each target experiment with the 'Portugal fifteen class' data set are listed in table 7-5 to table 7-10. The overall accuracies for each position in the *Rank* matrices will be analysed first as they provide a summary of the results for each target type. Since each target type was tested on the same data sets, using the same network parameters, the overall accuracies are directly comparable. Individual matrices can then be studied to establish the patterns of mis-identification of positions and mis-classification. Overall percentage accuracies for each position (vertically) and each target type (horizontally) for the *soft* classification of the 'Portugal fifteen class' data set are summarised in table 7-4. Figure 7-8 shows the same results in the form of a bar chart.

Position of the Mixture Component	Overall Percentage Classification Accuracy						Numbers of Pixels
	Pure	Scaled (-1,+1)	Scaled (0, +1)	Bin (6)	Bin (4)	Occur'	
Dominant	70.17	62.70	61.45	61.77	63.02	49.28	*18,434*
Secondary	26.18	39.27	39.90	41.97	**36.31**	40.42	*14,254*
Third	21.23	40.75	40.37	50.16	21.58	38.74	***8,182***
Fourth	37.13	30.13	29.33	34.80	22.60	23.41	*1,115*

Table 7-4. Overall percentage accuracies for each position in a mixture and each target type for the 'Portugal fifteen class' data set

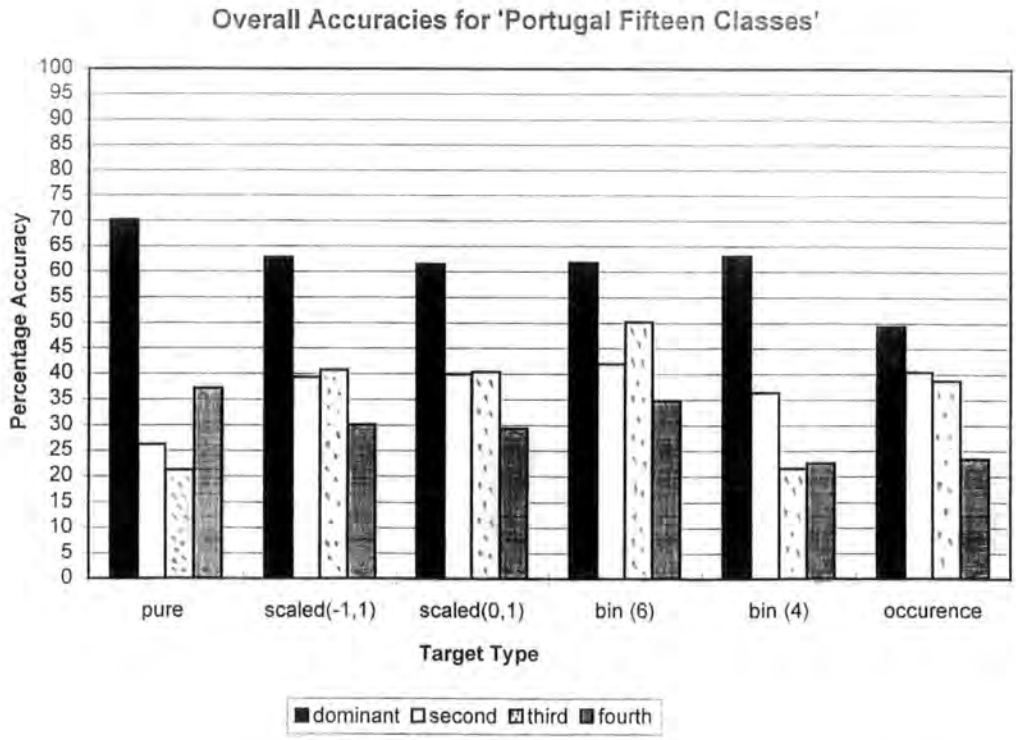


Figure 7-8. Comparison of overall accuracies for each target type and each position for the 'Portugal fifteen class' data set

#File used in confusion matrix: Portugal_15_Target_Pure

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	*True* Accuracy	Number of Pixels
1	12935	2732	1018	553	334	208	114	53	55	53	41	55	70	78	135	== 70.17%	- 70.17%	- (18434)
2	1596	3732	1938	1257	857	469	267	203	202	193	245	359	496	673	**1767**	== 26.18%	- 21.27%	- (14254)
3	448	1500	1737	796	619	359	202	166	140	110	132	229	331	467	946	== 21.23%	- 6.80%	- (8182)
4	146	177	129	414	73	35	15	15	19	30	26	19	8	5	4	== 37.13%	- 5.02%	- (1115)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	Number of Pixels
1	3826	0	0	55	350	368	723	327	0	1393	1062	1614	0	1424	759	== 32.15%	- (11901)
2	62	0	0	0	0	5	0	0	0	2	0	0	0	0	0	== 0.00%	- (69)
3	0	0	0	0	4	0	3	0	0	0	0	30	0	0	56	== 0.00%	- (93)
4	14	0	0	834	68	4	24	18	0	42	1	41	0	2	21	== 78.02%	- (1069)
5	0	0	0	5	73	0	0	2	0	31	0	1	0	0	5	== 62.39%	- (117)
6	39	0	0	0	12	427	13	21	9	118	29	86	0	11	18	== 54.53%	- (783)
7	8	0	0	0	0	5	166	0	0	246	27	68	0	2	95	== 26.90%	- (617)
8	16	0	0	0	12	2	38	104	0	176	24	62	0	7	4	== 23.37%	- (445)
9	18	0	0	0	29	0	0	7	0	18	11	1	0	1	3	== 0.00%	- (88)
10	791	0	0	30	287	311	60	82	10	7418	232	945	0	1028	414	== 63.90%	- (11608)
11	455	9	0	32	187	57	77	16	1	267	2009	357	0	623	109	== 47.84%	- (4199)
12	238	0	0	0	90	99	274	107	0	462	42	973	0	378	310	== 32.73%	- (2973)
13	28	0	0	2	36	7	5	25	0	36	14	44	9	48	25	== 3.23%	- (279)
14	351	1	0	0	168	84	91	97	0	709	210	1177	0	1385	575	== 28.57%	- (4848)
15	62	0	0	3	190	51	61	99	1	303	37	278	0	217	1594	== 55.04%	- (2896)

Table 7-5. *Rank* and *Modified* confusion matrices for the *Pure* target

#File used in confusion matrix: Portugal_15_Target_Scaled_Ones

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- *True* Accuracy	- Number of Pixels
1	11558	3379	1714	746	399	204	116	81	48	36	25	22	28	25	53	== 62.70%	- 62.70%	- (18434)
2	**2647**	**5598**	**2016**	**1333**	881	508	345	201	112	80	63	74	83	122	191	== 39.27%	- 32.59%	- (14254)
3	764	797	3334	1460	777	398	211	120	75	39	23	15	36	90	43	== 40.75%	- 23.22%	- (8182)
4	192	206	136	336	100	27	10	20	32	29	11	6	2	7	1	== 30.13%	- 0.00%	- (1115)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- Number of Pixels
1	4957	6	0	31	155	256	677	176	0	1277	1108	1720	0	857	681	== 41.65%	- (11901)
2	66	0	0	0	0	3	0	0	0	0	0	0	0	0	0	== 0.00%	- (69)
3	0	0	0	0	0	0	7	0	0	0	0	37	0	7	42	== 0.00%	- (93)
4	41	0	0	825	35	5	13	24	0	47	4	35	0	7	33	== 77.17%	- (1069)
5	7	0	0	0	78	0	0	2	0	23	0	1	0	4	2	== 66.67%	- (117)
6	33	0	0	0	1	399	16	8	0	141	16	137	0	22	10	== 50.96%	- (783)
7	19	0	0	0	0	3	223	2	0	246	4	17	0	2	101	== 36.14%	- (617)
8	27	0	0	0	1	0	61	133	0	143	23	40	0	13	4	== 29.89%	- (445)
9	25	0	0	0	17	2	0	6	0	21	14	1	0	1	1	== 0.00%	- (88)
10	825	7	0	34	31	108	92	33	0	6771	375	1321	0	1594	417	== 58.33%	- (11608)
11	625	6	0	10	5	17	9	9	0	246	2800	344	0	116	12	== 66.68%	- (4199)
12	44	0	0	0	49	136	92	21	0	291	20	1523	0	461	336	== 51.23%	- (2973)
13	48	0	0	0	1	20	1	5	0	32	23	72	4	63	10	== 1.43%	- (279)
14	420	0	0	0	5	15	40	50	0	973	98	1251	0	1669	327	== 34.43%	- (4848)
15	161	0	0	5	92	51	28	34	0	248	38	480	0	315	1444	== 49.86%	- (2896)

Table 7-6. *Rank* and *Modified* confusion matrices for the *Scaled (-1, 1)* target

#File used in confusion matrix: Portugal_15_Target_Scaled_Membership

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- *True* Accuracy	- Number of Pixels
1	11328	3518	1725	727	436	210	163	98	45	24	22	13	32	19	74	== 61.45%	- 61.45%	- (18434)
2	2700	5687	2153	1428	801	444	179	180	145	95	67	50	53	81	191	== 39.90%	- 31.73%	- (14254)
3	682	1193	3303	1480	632	318	129	115	86	58	34	23	24	40	65	== 40.37%	- 20.18%	- (8182)
4	220	118	149	327	90	48	43	13	37	44	15	7	0	4	0	== 29.33%	- 0.00%	- (1115)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- Number of Pixels
1	5132	0	0	27	152	214	514	126	0	1289	1062	1845	0	1077	463	== 43.12%	- (11901)
2	66	0	0	0	0	3	0	0	0	0	0	0	0	0	0	== 0.00%	- (69)
3	4	0	0	0	4	0	0	0	0	0	0	47	0	0	38	== 0.00%	- (93)
4	49	0	0	813	28	2	1	10	0	83	0	54	0	6	23	== 76.05%	- (1069)
5	8	0	0	0	74	0	0	8	0	26	0	1	0	0	0	== 63.25%	- (117)
6	37	0	0	0	0	385	4	19	0	161	13	144	0	9	11	== 49.17%	- (783)
7	35	0	0	0	0	1	74	0	0	325	2	6	0	0	174	== 11.99%	- (617)
8	40	0	0	0	2	3	1	89	0	235	20	39	0	11	5	== 20.00%	- (445)
9	31	0	0	0	14	0	0	6	0	13	12	12	0	0	0	== 0.00%	- (88)
10	1097	0	0	34	49	137	32	17	0	6963	304	1077	0	1537	361	== 59.98%	- (11608)
11	587	2	0	2	28	31	22	7	1	340	2657	352	0	169	1	== 63.28%	- (4199)
12	88	0	0	0	93	144	63	49	0	321	21	1470	0	411	313	== 49.45%	- (2973)
13	40	0	0	1	6	13	0	12	0	28	17	80	4	64	14	== 1.43%	- (279)
14	476	3	0	0	42	18	97	8	0	968	79	1268	0	1638	251	== 33.79%	- (4848)
15	200	0	0	6	147	58	22	11	0	185	38	548	0	335	1346	== 46.48%	- (2896)

Table 7-7. *Rank* and *Modified* confusion matrices for the *Scaled (0, 1)* target

#File used in confusion matrix: Portugal_15_Target_Bin_6

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	*True* Accuracy	Number of Pixels
1	11387	3629	1682	735	481	167	117	65	47	23	18	17	14	25	27	== 61.77%	61.77%	(18434)
2	2922	5983	2120	1284	799	345	182	136	102	88	56	45	87	63	42	== 41.97%	34.57%	(14254)
3	752	880	4104	1267	747	228	56	49	22	17	11	13	16	17	3	== 50.16%	28.01%	(8182)
4	189	133	150	388	97	52	47	25	14	10	4	1	0	0	5	== 34.80%	0.18%	(1115)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	Number of Pixels
1	**5362**	0	0	25	134	124	626	97	0	**1494**	**1116**	**1467**	0	918	538	== 45.06%	(11901)
2	65	0	0	0	0	2	0	0	0	2	0	0	0	0	0	== 0.00%	(69)
3	0	0	0	0	1	0	0	0	0	0	0	27	0	3	62	== 0.00%	(93)
4	32	0	0	846	42	0	9	16	0	53	0	25	0	10	36	== 79.14%	(1069)
5	7	0	0	0	85	0	0	2	0	17	0	1	0	4	1	== 72.65%	(117)
6	30	0	0	6	8	392	6	1	0	155	16	154	0	4	11	== 50.06%	(783)
7	53	0	0	0	0	0	71	1	0	335	0	62	0	0	95	== 11.51%	(617)
8	47	0	0	0	2	0	7	124	0	219	13	17	0	11	5	== 27.87%	(445)
9	37	0	0	0	0	0	0	6	0	19	23	1	0	1	1	== 0.00%	(88)
10	1077	3	0	43	40	86	55	26	0	7316	375	819	0	1406	362	== 63.03%	(11608)
11	688	3	0	3	0	18	10	13	0	291	2858	183	0	130	2	== 68.06%	(4199)
12	169	0	0	0	50	133	29	48	0	290	10	1524	0	426	294	== 51.26%	(2973)
13	35	0	0	0	7	9	2	20	0	20	11	114	14	43	4	== 5.02%	(279)
14	520	3	0	1	7	16	53	17	0	985	62	1149	0	1772	263	== 36.55%	(4848)
15	222	0	0	0	76	47	9	0	0	238	14	496	0	296	1498	== 51.73%	(2896)

Table 7-8. *Rank* and *Modified* confusion matrices for the *Bin 6* target

#File used in confusion matrix: Portugal_15_Target_Bin_4

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- *True* Accuracy	- Number of Pixels
1	11617	3548	1529	557	364	241	129	97	29	29	24	22	46	69	133	== 63.02%	- 63.02%	- (18434)
2	2262	5176	1705	890	760	524	356	275	209	172	151	176	382	526	690	== 36.31%	- 28.41%	- (14254)
3	755	1345	1766	711	669	487	442	290	159	116	142	143	296	448	413	== 21.58%	- 7.31%	- (8182)
4	352	165	111	252	51	31	16	12	9	9	13	39	34	21	0	== 22.60%	- 0.09%	- (1115)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- Number of Pixels
1	4290	0	0	46	211	327	661	167	0	1183	1198	2254	0	908	656	== 36.05%	- (11901)
2	67	0	0	0	0	0	0	0	0	2	0	0	0	0	0	== 0.00%	- (69)
3	0	0	0	0	3	0	3	0	0	0	0	28	0	4	55	== 0.00%	- (93)
4	32	7	0	830	35	0	20	36	0	44	1	13	0	5	46	== 77.64%	- (1069)
5	6	0	0	0	80	0	0	2	0	22	0	1	0	4	2	== 68.38%	- (117)
6	20	0	0	0	10	427	16	11	0	106	18	151	0	14	10	== 54.53%	- (783)
7	20	0	0	0	0	1	135	1	0	300	5	46	0	0	109	== 21.88%	- (617)
8	13	0	0	0	9	3	67	104	0	173	25	36	0	10	5	== 23.37%	- (445)
9	21	0	0	0	24	0	0	9	0	15	12	0	0	6	1	== 0.00%	- (88)
10	953	6	0	20	63	496	143	44	12	6418	421	1049	0	1503	480	== 55.29%	- (11608)
11	589	4	0	5	108	154	80	20	12	347	1971	557	0	322	30	== 46.94%	- (4199)
12	58	0	0	15	76	127	47	60	0	388	36	1499	0	354	313	== 50.42%	- (2973)
13	42	0	0	5	12	9	5	19	0	30	27	78	3	30	19	== 1.08%	- (279)
14	336	0	0	1	102	77	90	56	0	884	85	1155	0	1597	465	== 32.94%	- (4848)
15	114	4	0	0	192	104	24	96	0	127	34	398	0	346	1457	== 50.31%	- (2896)

Table 7-9. *Rank* and *Modified* confusion matrices for the *Bin 4* target

#File used in confusion matrix: Portugal_15_Target_Occurence

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- *True* Accuracy	- Number of Pixels
1	9084	4366	2547	1120	642	269	87	77	44	45	29	52	38	17	17	== 49.28%	- 49.28%	- (18434)
2	4343	5761	2521	704	495	148	97	44	20	14	25	21	24	11	26	== 40.42%	- 26.06%	- (14254)
3	1630	2150	3170	734	260	75	46	30	20	16	4	0	13	19	15	== 38.74%	- 13.15%	- (8182)
4	229	179	137	261	220	44	11	16	4	1	2	0	2	8	1	== 23.41%	- 0.27%	- (1115)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total Accuracy	- Number of Pixels
1	5448	0	0	45	23	29	87	14	0	2975	1321	539	0	1040	380	== 45.78%	- (11901)
2	67	0	0	0	0	1	1	0	0	0	0	0	0	0	0	== 0.00%	- (69)
3	1	0	0	0	0	0	5	0	0	4	0	4	0	0	79	== 0.00%	- (93)
4	69	0	0	864	49	3	0	4	0	40	2	2	0	14	22	== 80.82%	- (1069)
5	1	0	0	61	35	0	0	0	0	11	0	3	0	6	0	== 29.91%	- (117)
6	29	0	0	9	0	367	10	5	0	144	49	134	0	27	9	== 46.87%	- (783)
7	322	0	0	1	4	0	7	0	0	163	0	100	0	1	19	== 1.13%	- (617)
8	127	0	0	24	0	1	0	68	0	168	26	13	0	8	10	== 15.28%	- (445)
9	36	0	0	0	1	0	0	3	0	11	31	6	0	0	0	== 0.00%	- (88)
10	3072	0	0	91	8	13	5	7	0	5381	515	455	0	1654	407	== 46.36%	- (11608)
11	883	0	0	0	0	3	14	7	0	687	2244	64	0	294	3	== 53.44%	- (4199)
12	567	0	0	10	13	102	1	19	0	159	17	1155	0	421	509	== 38.85%	- (2973)
13	57	0	0	2	0	2	0	5	0	27	14	27	90	53	2	== 32.26%	- (279)
14	1106	1	0	2	2	8	2	4	0	1304	75	577	0	1538	229	== 31.72%	- (4848)
15	190	0	0	8	3	5	3	0	0	325	31	913	1	338	1079	== 37.26%	- (2896)

Table 7-10. *Rank* and *Modified* confusion matrices for the *Occurence* target

Overall accuracies

Table 7-4 shows that there are a maximum of four components in a mixture according to the reference data for this data set. The numbers of pixels containing at least each number of mixture components are listed in the last column on the relevant row. For example, there are 8,182 pixels with at least three components (shown in bold). As explained in section 7.1.2, the overall accuracies represent the number of times classes in a particular position are correctly identified. For example, according to the results in table 7-4, the network trained with the *Bin (4)* type of target correctly identifies the secondary class of 36.31% of the total number of pixels which contain at least two components (shown in bold).

The table and the bar chart show that overall levels of accuracy generally decrease from dominant to fourth component for all the target types and levels of accuracy are particularly low beyond the dominant class.

- The dominant class was identified best by the network using *Pure* targets (70.17%) and worst by the network using *Occurrence* targets (49.28%). Other target types produce accuracies that are similar to each other (~62%).
- The secondary classes are identified worst by the network trained with the *Pure* target (26.18%) and to a similar degree by the other targets (~40%).
- The networks using *Pure* and *Bin (4)* target types register the lowest accuracy of identification of the third components (~21%) and the *Bin(6)* target type the highest (50.16%); other target types are similar (~39%).
- The fourth component registers the highest accuracy for the network trained with *Pure* targets (37.13%) and lowest for the network trained with *Bin (4)* targets (22.60%).

If the network simply registered random values at its output nodes, each position would have a 1 in 15 chance of being assigned the correct class or ~7% accuracy (6.66%). Considering that overall accuracies are higher than 7%, it can be concluded that the network does not allocate classes randomly. If neural network output values were solely based on the frequency of classes, then the neural network would calculate an average output vector in the training phase and output values for all unseen pixels would be proportional to the number of times the classes occurred in the training set. This is not the case and so network classification cannot be based only on the frequency of classes. Furthermore, since accuracies for the same position differ between target types, the target type must have an influence.

The decrease in accuracy for each mixture component is to be expected. As percentage cover decreases, the contribution of the class with that cover to the overall pixel signal will probably decrease. For example, the second component in a [90% class x + 10% class y] mixture will be difficult for the network to identify. In addition, in situations where percentage cover is more or less equally divided between the components, for example [30% class 2 + 30% class 5 + 40% class 7], the order of the classes will be difficult to identify. Smaller percentages may also be more difficult to estimate visually and the error in the ground truth may be increased.

The *Pure* target was not expected to classify the dominant class accurately. In fact, the *Pure* target registers the highest accuracy for the dominant class of any target type. This suggests that the network may be particularly robust towards increased variability in class definition. The reason for which the *Pure* target performs better for the dominant class than the other target types may be connected to the other target types' representation of reference data. For example, 70% cover is represented by +0.4 in the *Scaled (-1,1)* type of target. The other *Scaled* and *Bin* target types are similar. Assuming that the network is able to assign the correct output value to the correct node for a pixel covered 70% with a class, the dominant class would only be misclassified in the following situations:

- for the *Pure* target, another output node must register a value higher than +1;
- on the other hand, for the *Scaled (-1,1)* target, another output node only needs to register a value greater than 0.4.

Furthermore, pixels which are covered by several classes whose individual cover does not exceed 50%, will have *Scaled* and *Bin* target values that only contain negative numbers or in the case of the *Scaled(0,1)* target type, only contain low target output values. These pixels may confuse the network as the spectral signal from each class may be weak and the network cannot use the full range of target values.

The *Scaled* and *Bin (6)* targets produce the highest accuracies of classification of the second and third components. This suggests that identifying component classes is helpful to the network. Nevertheless, classification accuracies are low. Surprisingly, the third components are sometimes identified as well as or better than the second components. This may simply be a result of fewer pixels containing at least three components.

The *Bin (4)* target perform less well for the third component than the other types of similar targets but better for the dominant class. This may have to do with its representation of reference data. In order for a class to be signalled by a value other than -1 for this target type, the class must cover an area larger than 25% of the pixel. This

means that for any pixel whose individual components do not cover more than 25% of the total area, -1 will be assigned to all the classes. These pixels will be of no use to the network. Furthermore, it is impossible to identify a fourth component with the *Bin (4)* target since the minimum percentage cover for a class to be allocated a value larger than -1 is 30% (since pixel percentage values are multiples of 10, see section 4.4.1) and it is not possible to have a four component mixture where the fourth component covers 30% of the area of the pixel. Therefore, it is not really surprising that the network trained with the *Bin(4)* target type performs less well for these components. On the other hand, a class only needs to cover 80% of the pixel for its target value to be +1 and the argument advanced for the *Pure* target applies.

The *Occurrence* target was not expected to perform well since the network is taught that class order is indifferent. However, it has relatively high accuracies in the second and third positions. This implies that the network seems to identify the correct classes that are present in a pixel. If the network is assumed to correctly identify component classes of a mixed pixel then the probabilities with which it will assign then in the right order can be calculated. For example, for a mixture of class 2 and class 8, it has a 1 in 2 chance of assigning the correct classes to the dominant and secondary positions. For a three component mixture, each position has a 1 in 3 chance of being assigned the correct class and so on. Consequently, always assuming that the network has actually learned which classes are present within the pixel, the percentage of all dominant classes being correctly identified is: 100% of *pure* pixels, 50% of two class pixels, 33% of three class pixels, 25% of four class pixels. Since overall accuracies are only calculated at each position, regardless of the order of classes before and after, the percentage of all secondary classes being correctly identified is 50% of two class pixels, 33% of three class pixels, 25% of four class pixels and so on.

For the 'Portugal fifteen class' data set used in these experiments, there are 1,115 pixels with four components; (8,182 - 1,115) = 7,067 only three component pixels; (14,254 - 8,182) = 6,072 only two component pixels and (18,434 - 14,254) = 4,180 *pure* pixels. So, the probability of correct classification of the dominant class, assuming that the network can identify the correct classes, is (4180 + 3,036 + 2,354 + 279) / 18,434 = ~53%; that for the second class is ~40%; that for the third class is ~32% and that for the fourth class is~ 25%. The results in table 7-4 are relatively similar to these values. The dominant class of 49.28% of the pixels, the secondary class of 40.42% of the pixels containing at least two components, the third class of 38.74% of the pixels containing at least three components and 23.41% of the pixels containing four components are correctly

identified. The differences in the results are probably a function of the fact that the neural network may not have learned the correct classes for each pixel and that different class frequencies may bias the network.

The fourth component is identified best by the *Pure* target. The reason for this is unclear since *Pure* targets provide no information about mixtures. If it is hypothesised that the network can extract spectral signatures from mixtures, having been trained on *Pure* classes, this does not explain why the network trained with this target type identified secondary and third components least well of all the targets. All fourth components cover a maximum of 20% of the pixel. It may be that the non *Pure* targets are confusing the network by, in a way, attaching too much importance to four component pixels. Pixels with four components will have approximately equally divided proportions between at least the third and fourth and often the secondary, third and fourth classes (for example, [70% class 6 + 10% class 2 + 10% class 1 + 10% class 10] or [30% class 5 + 20% class 3 + 20% class 1 + 20% class 4]). Either way, the fourth component is a minor proportion of the pixel and is expected to contribute little to the overall spectral signature. However, the *Scaled (-1,1)*, *Scaled (0,1)* and *Bin(6)* target types will provide the network with different targets for [70% class 6 + 10% class 2 + 10% class 1 + 10% class 10] and [70% class 6 + 20% class 2 + 10% class 1] even though it is unlikely that the spectral signatures differ very much.

True overall accuracies

The overall *true* accuracies are calculated by only considering a position as correctly classified if the previous position was also correctly identified, as explained in section 7.1.2. Thus, classification accuracies for second, third and fourth components are expected to reduce substantially. The corresponding *true* accuracies for each target type experiment described above are listed in table 7-11 and displayed in a bar chart in figure 7-9. The table shows, for example, that the network trained with *Pure* targets allocated the correct dominant and second class for 21.27% of pixels with at least two components (shown in bold) or that the network trained with *Bin (6)* targets allocated the correct first, second and third classes to 28.01% of pixels with at least three components (shown in bold).

As expected, the classification accuracies of secondary, third and fourth classes have decreased. The difference between the overall accuracies and the true overall accuracies for each target type and each position are shown in table 7-12. The accuracies of identification of secondary (21.27%) and third components (6.80%) are lowest for the

Position of the Mixture Component	Overall *True* Percentage Classification Accuracy						Numbers of Pixels
	Pure	*Scaled (-1,+1)*	*Scaled (0, +1)*	*Bin (6)*	*Bin (4)*	*Occur'*	
Dominant	70.17	62.70	61.45	61.77	63.02	49.28	*18,434*
Secondary	**21.27**	32.59	31.73	34.57	28.41	26.06	*14,254*
Third	6.80	23.22	20.18	**28.01**	7.31	13.15	*8,182*
Fourth	5.02	0.00	0.00	0.18	0.09	0.27	*1,115*

Table 7-11. Overall *true* accuracies for each position in a mixture and each target type for the 'Portugal fifteen class' data set

Position of the Mixture Component	Difference between overall and *true* overall accuracies						Numbers of Pixels
	Pure	*Scaled (-1,+1)*	*Scaled (0, +1)*	*Bin (6)*	*Bin (4)*	*Occur'*	
Secondary	-4.91	-6.68	-8.17	-7.4	-7.9	-14.36	*14,254*
Third	-14.43	-17.53	-20.19	-22.15	-14.27	-25.59	*8,182*
Fourth	-32.11	-30.13	-29.33	-34.62	-22.51	-23.41	*1,115*

Table 7-12. Difference between overall and *true* overall accuracies for the 'Portugal fifteen class' data set

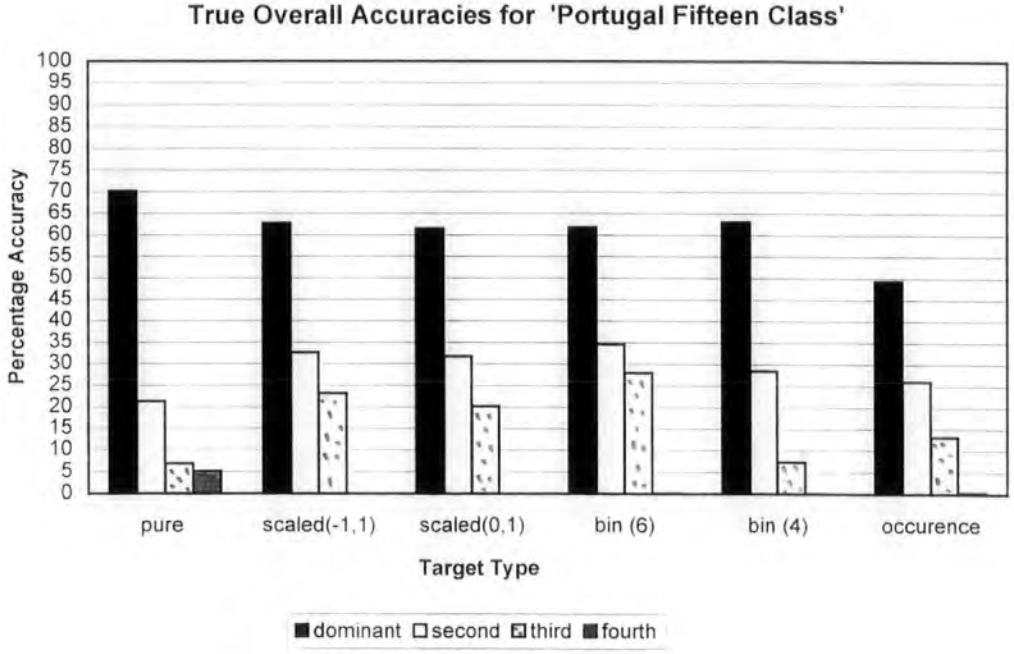


Figure 7-9. Comparison of *true* overall accuracies for each target type and each position for the 'Portugal fifteen class' data set

Pure target type. The change in accuracies for the secondary and third components is largest for the *Occurrence* target. For example, for the secondary classes, whereas the other targets average a change of -7%, the *Occurrence* target registers a change of -14.36%. The *Occurrence* target results for secondary (26.06%) and third components (13.15%) are now greater than those of the *Pure* target type but less than the accuracy values of the other target types (approximately 32% for the secondary class and approximately 24% for the third class). The identification of third components (7.31%) by the network trained with *Bin (4)* targets is an exception as its value is between the *Pure* and *Occurrence* values.

The results suggest that providing information about the components of pixels is helpful to the network since *Pure* targets, which do not provide any mixture information perform worst for the secondary and third classes. Furthermore, information which suggests the ranking order of classes is of more value to the network than simply indicating the presence of classes since *Occurrence* targets perform worse than the other types of targets for the secondary and third components. The definition of targets has an effect on the classification accuracy since *Scaled* and *Bin* targets register different accuracies. There is little difference between the two types of *Scaled* targets. It would seem that information that is too precise may confuse the network since *Scaled* targets perform worse than *Bin* (6) targets. However, range definitions must be carefully analysed since the *Bin (4)* target type performs better than the *Bin (6)* target in the dominant class but worse that the *Scaled* and *Bin (6)* target type in the secondary and third classes.

For each position, the change in accuracy is approximately similar for most target types. It can be speculated that it may be the same set of pixels whose order of classes is being wrongly interpreted. Perhaps it is mainly those pixels whose area is divided up approximately equally between classes or pixels whose reference data is wrong. Unfortunately, these hypotheses cannot be verified due to the loss of locational information at the stage of creating the data sets and because of the complexity of the data sets.

First four components

Having compared the overall and *true* overall classification accuracies for the different target types, the mixing information provided by the individual matrices can be examined. The *Rank* matrices listed in table 7-5 to table 7-10 show that for all the target types, frequencies of allocation tend to decrease as the position according to the neural network increases from 1 to 15. In fact, the position to which a class is allocated to when

it is mis-assigned seems to be more frequently one of the first four components than any other position. Indeed for most targets each position is most frequently mis-assigned to the position either immediately below or above it.

For example, in table 7-6, which provides the results for the *soft* classification of the 'Portugal fifteen class' data set using targets of the type *Scaled (-1,1)*, the secondary class is correctly identified in 5,598 pixels (shown in bold) out of 14,245 pixels with at least two components, which is equivalent to 39.27%. The most common individual mis-allocations of secondary classes for this target are to the first (2,647) and to the third (2, 016) positions, followed by to the fourth (1,333) position (shown in bold). Together, these account for 81,3% of the pixels which contain at least two components.

The network trained with the other target types also allocate most mis-identified classes to the first four components. However, the *Pure* target shown in table 7-5 registers the third highest individual mis-allocation to the 15^{th} position (shown in bold) for the secondary classes (1,767 pixels). In fact, the *Pure* target shows a greater distribution of allocations over all the possible network positions than the other target types. The other target types tend to have more mis-allocations concentrated in the first few components. This is with the exception of the *Bin (4)* target, shown in table 7-9, which also shows a greater distribution of allocations over all the possible network positions for the secondary and third components.

The percentage accuracy if assignment of the components of a pixel to any of the first four positions was considered correct are calculated from the *Rank* matrices for each target type and each position. Table 7-14 lists the values and figure 7-9 presents the same results in bar chart format. As the table and the figure show, the large majority of assignments are within the first four positions. For example, the dominant classes for 93.8% of pixels were assigned as the dominant, second, third or fourth components by the neural network for the *Scaled (0,1)* target (shown in bold). Accuracies of allocation have increased to ~94% for the dominant class, an average of ~80% for the second and ~75% for the third and fourth components. This suggests that for the majority of pixels the network is able to identify the components of the pixel, since these accuracies are high, but that it has difficulty ordering the components, since the individual accuracies at each position are low, as shown in the previous section.

Using this measure of accuracy, the target type with the highest accuracy of identification of the dominant class is the *Bin (6)* target type (94.6%) although there is little difference between the targets. The second and third components are best identified using the *Occurrence* target type (93.5% and 93.9% respectively) whereas the *Pure* target

Position of the Mixture Component	Overall Percentage Classification Accuracy						Numbers of Pixels
	Pure	*Scaled (-1,+1)*	*Scaled (0,1)*	*Bin (6)*	*Bin (4)*	*Occur'*	
Dominant	93.5	94.4	**93.8**	94.6	93.6	92.9	*18,434*
Secondary	59.8	81.3	84.0	86.4	70.4	93.5	*14,254*
Third	54.8	77.7	81.4	85.6	55.9	93.9	*8,182*
Fourth	77.7	78.0	73.0	77.1	78.9	72.3	*1,115*

Table 7-13. Overall percentage accuracies if assignment to any of the first four components is considered correct

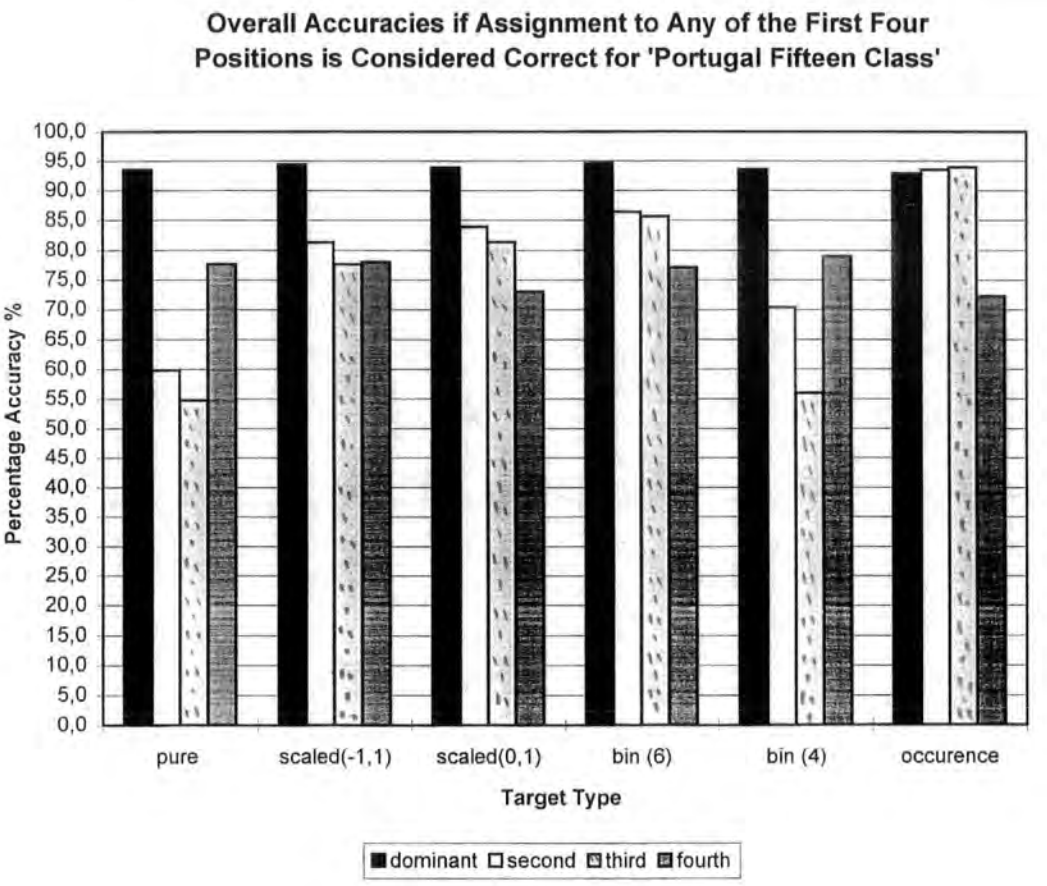


Figure 7-10. Comparison of overall percentage accuracies if assignment to any of the first four components was considered correct

type has the lowest accuracy for identifying these components (59.8% and 54.8% respectively). The *Scaled* and *Bin (6)* target types have accuracies of (~84%) for the secondary components and (~82%) for the third components. The *Bin (4)* target registers values of 70.4% for the secondary and 55.9% for the third components. The fourth component is identified with the highest degree of accuracy by the *Bin (4)* target type (78.9%) and with the lowest level of accuracy by the *Occurrence* target (72.3%).

If the percentage of correctly identified positions is calculated as a function of the number of pixels in only the first four positions according to the neural network allocations, this provides an indication of the distribution of pixels within those positions. The results are shown in table 7-14 and figure 7-11. The *Occurrence* target, which had the highest overall accuracies in the previous table, is shown to distribute class allocation more randomly between the first four positions since its accuracy of the dominant class over the four positions is now the lowest, 53.1% compared to an average of ~66% for the *Scaled* and *Bin* targets and 75.0% for the *Pure* target. This is to be expected since this target type is not taught any order for the classes. The levels of accuracy for the secondary and third components now average ~47 for both. The fourth component averages ~39%.

The results show that the neural network seems able to identify the components of pixels but not to rank them in the correct order. Identifying the component classes and their percentage cover is helpful to the network since secondary and third classes are identified best by *Scaled* and *Bin* target types. However, the target definitions also introduce an element of confusion since the *Occurrence* target achieves the highest accuracy of identification when the order of the classes is considered irrelevant.

Modified confusion matrix

The *Modified* misclassification matrices in table 7-5 to table 7-10 show that the distribution of misclassifications is similar for all the target types. For example, for all the target types, **Bare Soil** (class 1) is usually most misclassified with **Grass** (class 10), **Vines** (class 11) and **Shrubs** (class 12). In table 7-8 which shows the matrices for the network trained with *Bin (6)* targets, **Bare Soil** is correctly identified for 5,362 pixels out of 11, 901 (= 45.06%) pixels which contain some **Bare Soil**. The single highest individual mis-allocations (shown in bold) are then to **Grass** (1,494 = 12.55%), **Shrubs** (1,467 = 12.34%) and **Vines** (1,116 = 9.37%).

These misclassification were expected since these classes were shown to have similar spectral signatures in chapter IV and **Shrubs** and **Vines** are intimate mixtures of plants and **Bare Soil**. The misclassifications of **Bare Soil** are then to **Stubble** (class 7),

Position of the Mixture Component	Overall Percentage Classification Accuracy						Numbers of Pixels
	Pure	*Scaled (-1,+1)*	*Scaled (0,1)*	*Bin (6)*	*Bin (4)*	*Occur'*	
Dominant	75.0	66.4	65.5	65.3	67.3	53.1	*18,434*
Secondary	43.8	48.3	47.5	48.6	51.6	43.2	*14,254*
Third	38.8	52.5	49.6	58.6	38.6	41.3	*8,182*
Fourth	47.8	38.6	40.2	45.1	28.6	32.4	*1,115*

Table 7-14. Overall percentage accuracies as a function only of the first four components

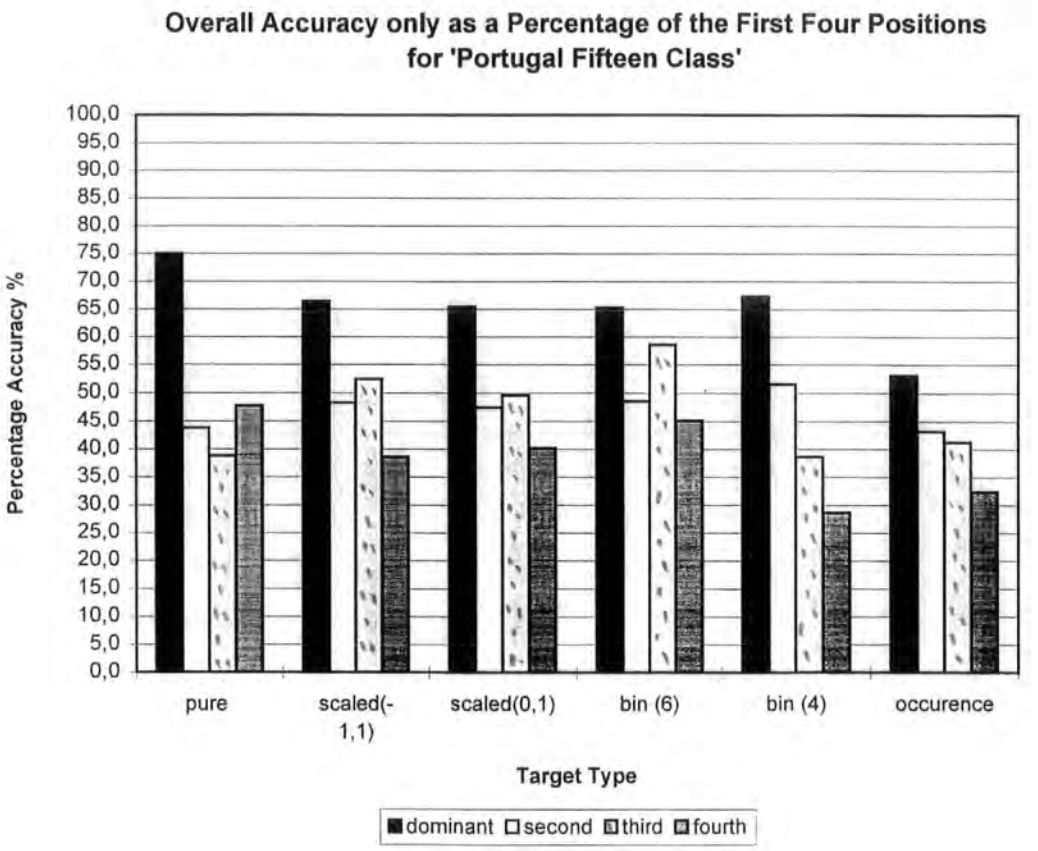


Figure 7-11. Comparison of overall percentage accuracies as a function only of the first four components

Broadleaf (class 14) and **Coniferous** (class 15). The **Stubble** class is an intimate mixture of plants and **Bare Soil**. The forest classes either contain some **Bare Soil** or are similar to the **Shrub** class which is itself confused with **Bare Soil**.

A well defined class such as **Water**, classified with 79.14% for this target type, is misclassified with **Marsh** and **Grass** mainly, followed by **Broadleaf** and **Bare Soil**. The misclassification with **Marsh** is understandable. It is likely that the misclassification with **Grass** occurs because **Marsh** can be misclassified with **Grass** and that the misclassification with **Broadleaf** occurs because that class is confused with **Grass** for some pixels.

Classes such as **Fruit Trees** and **Vines** were expected to be misclassified because they never occupy 100% of a pixel. However, **Vines** is relatively well identified whereas **Fruit Trees** is not. **Fruit Trees** is most misclassified with **Grass**. Since the **Fruit Trees** are either intimate mixtures with **Bare Soil** or **Grass** this is not surprising.

These simple examples suggest that the misclassification between classes is not unexpected and that the class definitions of some of the training samples, particularly **Bare Soil** and **Grass** may be poor. The problems of calculating percentage cover were illustrated in chapter IV, section 4.5, with the example of different **Vines** percentage cover.

7.2.3 Results and analysis: 'Portugal seven class' data set

For comparison purposes, the 'Portugal seven class' data set was also classified using networks with parameters set to the values in table 7-15, and a subset of the target types discussed above, namely *Pure*, *Scaled (-1,1)* and *Bin(6)*. This data set is chosen because it has fewer classes than the 'Portugal fifteen class' data set, only two component mixtures and no 50-50 mixtures and the results are expected to improve.

Parameters	Portugal 7 class
Number of iterations	600
Architecture: Input Nodes **Hidden Nodes** **Output Nodes**	6 13 7
Type of weight initialisation	Constant seed value
Learning rate	0.1
Randomisation	Once
Division ratio	TR : TS = 2 : 1 = 3,951 : 7,901

Table 7-15. Parameters for the neural networks using different target types for the 'Portugal seven class' data set

The targets for each composition within the 'Portugal seven class' data set are listed in table 7-16. As can be seen from this table, the targets for the *pure* pixels are the same for all the target types but differ for the mixed pixels. The compositions information for the *Bin (6)* is less variable than that for the *Scaled (-1,1)* target types since it assigns ranges of percentage cover to the same value.

Composition	Pure	Scaled (-1,1)	Bin (6)	Num.
100% 01 **Bare Soil**	+1 -1 -1 -1 -1 -1 -1	+1 -1 -1 -1 -1 -1 -1	+1 -1 -1 -1 -1 -1 -1	*984*
100% 02 **Sand**	-1 +1 -1 -1 -1 -1 -1	-1 +1 -1 -1 -1 -1 -1	-1 +1 -1 -1 -1 -1 -1	*102*
100% 03 **Water**	-1 -1 +1 -1 -1 -1 -1	-1 -1 +1 -1 -1 -1 -1	-1 -1 +1 -1 -1 -1 -1	*1196*
100% 04 **Urban**	-1 -1 -1 +1 -1 -1 -1	-1 -1 -1 +1 -1 -1 -1	-1 -1 -1 +1 -1 -1 -1	*528*
100% 05 **Cereals**	-1 -1 -1 -1 +1 -1 -1	-1 -1 -1 -1 +1 -1 -1	-1 -1 -1 -1 +1 -1 -1	*372*
100% 06 **Grass**	-1 -1 -1 -1 -1 +1 -1	-1 -1 -1 -1 -1 +1 -1	-1 -1 -1 -1 -1 +1 -1	*2827*
100% 07 **Vines**	-1 -1 -1 -1 -1 -1 +1	-1 -1 -1 -1 -1 -1 +1	-1 -1 -1 -1 -1 -1 +1	*1024*
20% 01 80% 06	-1 -1 -1 -1 -1 +1 -1	-0.6 -1 -1 -1 -1 +0.6 -1	-0.6 -1 -1 -1 -1 +0.6 -1	*1,063*
30% 01 70% 06	-1 -1 -1 -1 -1 +1 -1	-0.4 -1 -1 -1 -1 +0.4 -1	-0.2 -1 -1 -1 -1 +0.2 -1	*225*
40% 01 60% 06	-1 -1 -1 -1 -1 +1 -1	-0.2 -1 -1 -1 -1 +0.2 -1	-0.2 -1 -1 -1 -1 +0.2 -1	*248*
70% 01 30% 06	+1 -1 -1 -1 -1 -1 -1	+0.4 -1 -1 -1 -1 -0.4 -1	+0.2 -1 -1 -1 -1 -0.2 -1	*270*
80% 01 20% 06	+1 -1 -1 -1 -1 -1 -1	+0.6 -1 -1 -1 -1 -0.6 -1	+0.6 -1 -1 -1 -1 -0.6 -1	*714*
20% 06 80% 07	-1 -1 -1 -1 -1 -1 +1	-1 -1 -1 -1 -1 -0.6 +0.6	-1 -1 -1 -1 -1 -0.6 +0.6	*1419*
40% 06 60% 07	-1 -1 -1 -1 -1 -1 +1	-1 -1 -1 -1 -1 -0.2 +0.2	-1 -1 -1 -1 -1 -0.2 +0.2	*718*
20% 03 80% 06	-1 -1 -1 -1 -1 -1 +1	-1 -1 -0.6 -1 -1 +0.6 -1	-1 -1 -0.6 -1 -1 +0.6 -1	*162*

Table 7-16. Targets for each composition in the 'Portugal seven class' data set

Rank and *Modified* confusion matrices for each target type are in table 7-17 to table 7-19. Overall *true* accuracies are compared in table 7-20 The *true* accuracies of classification of the dominant and secondary components for the 'Portugal fifteen class' data set by neural networks trained with *Pure*, *Scaled (-1,1)* and B*in (6)* target types are listed in the first three data columns of table 7-20. The *true* accuracies of classification of the dominant and secondary cases for the 'Portugal seven class' data set, for a network trained with the same target types, are listed in the last three data columns of table 7-20.

The comparison between the true accuracies shows that the composition of pixels is estimated more accurately for the 'Portugal seven class' data set than the 'Portugal fifteen class' data set. The accuracy of identification of the dominant class averages ~84% for the 'Portugal seven class' data set whereas it averages ~65% for the 'Portugal fifteen class' data set. The accuracy of identification of secondary and dominant class for the 'Portugal seven class' data set averages ~54% whereas it average ~29% for the 'Portugal fifteen class' data set. Therefore, the networks perform better with the simpler data set.

#File used in confusion matrix: Portugal_7_Target_Pure.index

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	Total Accuracy	- *True* Accuracy	- Number of Pixels
1	6730	603	354	99	59	26	30	== 85.18%	- 85.18%	- (7901)
2	160	1799	734	151	150	161	49	== 56.15%	- 53.71%	- (3204)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	Total Accuracy	- Number of Pixels
1	1501	0	0	362	1	96	349	== 65.01%	- (2309)
2	53	1	0	15	0	0	0	== 1.45%	- (69)
3	53	0	803	14	2	31	9	== 88.05%	- (912)
4	8	0	0	268	0	79	8	== 73.83%	- (363)
5	14	0	0	119	0	112	8	== 0.00%	- (253)
6	416	0	1	450	5	4021	193	== 79.06%	- **(5086)**
7	45	0	0	20	0	113	1935	== 91.58%	- (2113)

Table 7-17. *Rank* and *Modified* confusion matrices for the *Pure* target classification of the 'Portugal seven class' data set

#File used in confusion matrix: Portugal_7_Target_Ones.index

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	Total Accuracy	- *True* Accuracy	- Number of Pixels
1	6587	792	318	106	59	18	21	== 83.37%	- 83.37%	- (7901)
2	524	1483	796	270	72	17	42	== 46.29%	- 43.29%	- (3204)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	Total Accuracy	- Number of Pixels
1	1747	0	13	189	19	205	136	== 75.66%	- (2309)
2	58	0	0	7	0	4	0	== 0.00%	- (69)
3	35	0	794	52	3	24	4	== 87.06%	- (912)
4	12	4	0	261	0	82	4	== 71.90%	- (363)
5	0	0	0	78	60	111	4	== 23.72%	- (253)
6	953	3	24	201	52	3597	256	== 70.72%	- (5086)
7	75	0	0	20	14	393	1611	== 76.24%	- (2113)

Table 7-18. *Rank* and *Modified* confusion matrices for the *Scaled (-1,1)* target classification of the 'Portugal seven class' data set

#File used in confusion matrix: Portugal_7_Target_Bin_6.index

RANK confusion matrix

pos_id	1	2	3	4	5	6	7	Total Accuracy	*True* Accuracy	Number of Pixels
1	6572	841	307	72	61	37	11	== 83.18%	83.18%	(7901)
2	494	2217	276	135	17	9	56	== 69.19%	66.54%	(3204)

Modified MISCLASSIFICATION confusion matrix

class_id	1	2	3	4	5	6	7	Total Accuracy	Number of Pixels
1	1617	0	0	204	34	354	100	== 70.03%	(2309)
2	50	0	0	17	0	2	0	== 0.00%	(69)
3	21	0	787	25	3	76	0	== 86.29%	(912)
4	1	0	0	264	0	94	4	== 72.73%	(363)
5	16	0	0	38	78	112	9	== 30.83%	(253)
6	251	0	1	299	11	4269	255	== 83.94%	(5086)
7	59	0	0	1	0	279	1774	== 83.96%	(2113)

Table 7-19. *Rank* and *Modified* confusion matrices for the *Bin (6)* target classification of the 'Portugal seven class' data set

	Overall *True* Percentage Classification Accuracy						
	Portugal 15 class			**Portugal 7 class**			
Pos.	*Pure*	*Scaled (-1,+1)*	*Bin (6)*	*Pure*	*Scaled (-1,+1)*	*Bin (6)*	**Num.**
Dominant	70.17	62.70	61.77	85.18	83.37	83.18	*7,901*
Secondary	21.27	32.59	31.73	53.71	43.29	66.54	*3,204*

Table 7-20. Overall *'true'* accuracies for each position in a mixture and each target type for the 'Portugal seven class' data set

In the experiments using the 'Portugal seven class' data set, the dominant class was best identified by the network using the *Pure* target type (85.18%) although there is little difference with the other two target types, *Scaled (-1,1)* and *Bin(6)*, which registered values of 83.37% and 83.18% respectively. These results are similar to those obtained with networks trained and tested on the 'Portugal fifteen class' data set. On the other hand, with this data set, the composition of the mixed pixels was identified worst by the *Scaled (-1,1)* target type (43.29%). The *Pure* target achieved 53.71% accuracy and the *Bin(6)* target type achieved 66.54%.

These results suggest that classification accuracy is not only a function of the target type but also of the actual composition of the data set. Here, all mixture contain **Grass** and most contain **Bare Soil**. Over half the pixels in the data set (5,086) contain **Grass** as can be seen from the matrices (for example table 7-17 in bold). The network trained with *Pure* target is therefore likely to assign a higher node value to any unseen pixel for the **Grass** class or the **Bare Soil** class and most of the time it will be correct, hence the relatively high accuracy of classification of the two component mixtures. In addition, the variability of the targets for the *Scaled (-1,1)* target type appears to confuse the network. The *Bin(6)* target type seems to achieve the right balance between providing the network with compositional information and not confusing it.

7.2.4 Summary

In summary, the experiments using the 'Portugla fifteen class' data set have shown that different target types influence classification accuracies. It seems that providing information about the composition of pixels is helpful to the network since *Scaled* and *Bin* targets generally identify secondary and third classes better than *Pure* or *Occurrence* targets. However, the information that is provided also seems to be the source of additional confusion. Therefore, it would seem that target types must be defined so as to strike the right balance between providing the compositions information and not adding to the possibility of classes being confused.

In the literature, Moody *et al.* (1996) carry out a classification of a testing data set containing mixed pixels using a network trained with a *Pure* target. They identify the second maximum value of the neural network output vector and assign the class with the second maximum value to the dominant class, for pixels that were mis-classified and find that classification accuracy increases. This suggests that the network assigns the correct class either as the dominant or the secondary class.

Results using the 'Portugal seven class' data set show that the overall *true* classification accuracies of the dominant and secondary classes of the 'Portugal seven class' data set are higher than those of the 'Portugal fifteen class data set'. Furthermore, the *Pure* target identifies the composition of two component pixels more accurately than the *Scaled (-1,1)* target. This suggests that the composition of the data sets may have a strong influence on classification accuracies.

7.3 Summary of Chapter VII

The limitations of existing methods to evaluate *soft* classification accuracies led to the development of two matrices for analysing neural network outputs. The *Rank* matrix, discussed in section 7.1.2, analyses the frequency with which classes in each position (dominant, second and so on) have been correctly identified and, if mis-identified, the position to which the network assigned them. The *Modified* confusion matrix, discussed in section 7.1.3, calculates the accuracies of classification of individual classes, regardless of the position they are in.

Using the matrices developed, the effect that different target types may have on classification accuracy was investigated. The experiments discussed in section 7.2 revealed that the network appears to be able to identify the components of pixels, that is the classes which are present within the pixel, but that it has difficulty ordering the classes correctly, that is determining which is the dominant class, the secondary class and so on. For the large majority of pixels, the network assigns the correct classes to one of the first four positions. Accuracies are ~90% for the dominant class, ~80% for the second and ~70% for the third and fourth classes if assignment of the correct class to any of the first four positions is considered to be correct. However, when *true* accuracies are calculated, that is when a position is only considered to be correct if the previous position was correctly identified, the overall percentage accuracies fall to ~60% for the dominant class, ~30% for the second, ~20% for the third and ~0% for the fourth component.

In fact, network trained with targets which specifically impose an order register higher accuracies in the second and third components than networks trained with targets that impose no order information. However, the network trained with a target which only signals the presence of a class, without indicating its order, registers the highest accuracy for identifying the component classes. This suggests that targets must strike a balance between providing enough information for the network to be able to order classes correctly and not provide information that is so detailed that the network has more probability of being wrong.

Results from the classification of the 'Portugal seven class' data set show that the overall *true* classification accuracies for the 'Portugal seven class' data set are higher than for the 'Portugal fifteen class data set'. The composition of the data sets appear to have a strong influence on classification accuracies since the network trained with the *Pure* target identifies the composition of two component pixels more accurately than the network trained with the *Scaled (-1,1)* target.

The *Pure* target generally produced lower accuracies than other target types for the

second and third components. This suggests that the network is not able to extract sub-pixel signatures and instead needs to be taught a pattern for mixed pixels. This is investigated further in the next, and final experiments chapter, using a data set with only two classes and very accurate ground data.

Chapter VIII

COMPOSITION OF TRAINING DATA SETS

The previous experiments, described in chapters VI and chapter VII, have suggested that network output values may be related to proportions of classes within pixels. Although experiments with different target types in chapter VII using the 'Portugal fifteen class' data set were inconclusive, the experiments with the 'Portugal seven classes' data set showed that ~50% of pixels could be correctly assigned dominant and secondary classes in the right order. It is thought that the size and complexity of the data set, although simpler than the fifteen class data set, may be the cause of the low accuracies.

In chapter VI, section 6.3, *Correspondence* images for the network trained and tested with mixed pixels from the 'Portugal seven classes' data set and a *Pure* target type showed that secondary classes seemed to be identified for some pixels. The node response graphs in section 6.4 for the network trained on *pure* classes from the Scotland site and tested on mixed pixels, showed that it produced the correct network node responses for several of the mixed pixels, that is high response for the appropriate classes and low responses everywhere else. It is interesting to find out whether the network requires examples of mixed pixels in order to accurately identify testing set mixed pixel compositions or whether the network requires *pure* pixels, from whose signatures it then derives mixed pixel compositions. If the network only requires *pure* pixels, then elaborate ground data collection exercises to accurately measure percentage cover of classes may no longer be required. On the other hand, it may be difficult to obtain *pure* pixels from some low resolution satellite imagery such as that from the AVHRR.

As mentioned above, the size and complexity of the data sets from Portugal and possible errors such as those described in chapter IV, section 4.5.2, are thought to be partly the cause of low levels of accuracy. For this reason, the structural data sets from Scotland were created. As discussed in chapter IV, structural parameters of a forest such as Height and Basal Area have been found to be strongly correlated with forest reflectance values from Landsat TM imagery (Butera, 1986). The correlation coefficient for the 'Scotland bio box' data set between Height and Landsat TM band 2, for example, was -

0.80 and that between Basal Area and band 2 was -0.74. In fact, Height and Basal Area are actually related to tree growth which itself is linked to canopy cover. The higher reflectance values for younger trees are thought to be attributable to the reflectance from ground vegetation such as grass. As canopy closure occurs, the relationship breaks down (White *et al.*, 1995). Therefore, since Height and Basal Area are in effect a measure of how much background can be seen, it is hypothesised in this thesis that they can be used as surrogates for class proportions for forest stands which have not reached canopy closure. The advantage of using such measures is that they can be measured on the ground very accurately.

The aim of the work described in this chapter is to determine whether the neural network can extract the relationship between Height and Basal Area measurements and Landsat TM channel values, and consequently, proportions of **Trees** and **Grass**. A review article by Kimes *et al.* (1998) suggests that the non-linearity of the neural network makes it particularly appropriate for this type of problem. This chapter aims to establish whether the network requires examples of mixed pixels in the training set in order to effectively predict values for pixels in a testing set, or whether the network can use only the *pure* end-members in the training file to establish some type of regression estimate with which to predict values for pixels in a testing set. As discussed in chapter II, the term 'end-members' is typically used in unmixing models to signify *pure* components, or composing components of a mixture.

This chapter is divided into two main sections. The first set of experiments require the network to invert the relationship between Landsat TM values and Height and Basal Area values; that is, the network is trained with the scaled measurements of Height and Basal Area. The targets are therefore continuous, in the sense that they are real values. This is in contrast with previous targets such as the *Pure* target which is a binary type of target. In a way, the scaled measurements are similar to the *Scaled* targets presented in the previous chapter except that there is only one node and it does not refer to a class but to a variable. The second set of experiments uses the same data sets but targets are no longer real continuous values; instead binary type targets are used. Preliminary work relevant to this chapter was described in Bernard and Donoghue (1996).

8.1 Correlation between Continuous Target and Network Outputs

The aim of these experiments is, using a continuous value as a target, to ascertain whether the network correctly estimates Height and Basal Area values for unseen pixels. If the network is capable of doing this, it can be assumed that it is recognising a

relationship between proportions of **Trees** and **Grass** on the ground and the Landsat TM signal. In addition, the experiments serve to investigate the requirements for the network to interpolate between data. Can the network be trained only with *pure* classes, in this case **100% Grass** and **100% Trees**, and when tested on mixed pixels correctly identify the components of the pixel and their proportions, or must the network be provided with examples of mixed pixels in the training data set in order to correctly identify the components of mixed pixels in the testing data set?

8.1.1 Methodology

The experiments were carried out on the structural data sets 'Scotland bio all' and 'Scotland bio box'. As discussed in chapter IV, section 4.8, graphs of the mean reflectance values for each band for increasing square kernels for the 'Scotland bio all' data set, which contains all the pixels from the digitised polygons, showed some variability within the data (appendix B, section B.8). For this reason, the 'Scotland bio box' data set was created which only contains pixels within a 5x5 box. Its graphs showed high homogeneity for pixels within the 5x5 box. Both data sets include the same *pure* **Grass** pixels. The 'Scotland bio box' data set has less variability than the 'Scotland bio all' data set and is expected to produce more accurate results.

The targets for the networks were Height or Basal Area values scaled between -1 and +1. In order to perform the scaling of the data, integer minimum and maximum values of both variables for each data set were identified from the reference data (chapter IV, section 4.6): 0 and 72 for Basal Area and 0 and 20 for Height. The test for the neural network was to estimate these scaled values of Height and Basal Area for unseen pixels.

For each of the Scotland data sets, two pairs of training and testing files were created.

- To test whether the network could approximate Height and Basal Area values having been trained with a file including mixed pixels, pixels were randomised and then divided in a 1:2 ratio between training and testing files. This data set is referred to as the '**representative**' data set since training pixels should be representative of the testing pixels.
- To test whether the network could approximate Height and Basal Area values having been trained with a file which did not include mixed pixels, polygons with complete canopy closure were identified and their pixels placed in a training file including the **Grass** pixels. Pixels from the other polygons, containing pixels of mixed **Trees** and **Grass**, were placed in the testing file.

This data set is referred to as the '**end-member**' data set because pixels in the training file are *pure* examples of the components of the pixels in the testing file.

The parameter settings for the network for each data set are listed in table 8-1. The same settings were used for the 'Scotland bio all' and the 'Scotland bio box' data sets. An architecture of four nodes in the hidden layer was deemed sufficient for adequate approximation.

The ability of the network to approximate the variable values is evaluated in graphs of neural network estimated Height and Basal Area values against reference data Height and Basal Area values. A perfect linear correlation would be shown by a 1:1 relationship. Neural network output values were post-processed by scaling them between 0 and 72 for the Basal Area and 0 and 20 for the Height; since this is post-process scaling it has no effect on the algorithm or the results.

Parameters	Representative	End-Member
Number of iterations for training	600	600
Architecture of the network: Input Nodes **Hidden Nodes** **Output Nodes**	6 4 1	6 4 1
Type of weight initialisation	Constant seed value	Constant seed value
Learning rate	0.1	0.1
Randomisation	Once	Once
Division ratio 'Scotland bio all'	1 : 2 = 312 : 624	(N/A) 367 : 569
Division ratio 'Scotland bio box'	1 : 2 = 172 : 343	(N/A) 291 : 226

Table 8-1. Parameter settings for the experiments with the 'representative' and 'end-member' data files for the 'Scotland bio all' and the 'Scotland bio box' data sets and the continuous target type

8.1.2 Results and analysis for the representative data sets

Figure 8-1 and figure 8-2 show the neural network estimations of Height and Basal Area respectively, for the **representative** testing file of the 'Scotland bio all' data set. A linear regression was fitted through the points for each graph whose equation is provided at the top of the graphs. R values which show the strength and direction of the correlation, and R^2 values which provides a measure of the explained variance were also calculated. For the Height for this data set, $R = +0.92$, $R^2 = 0.85$; for the Basal Area for this data set, $R = +0.89$, $R^2 = 0.79$.

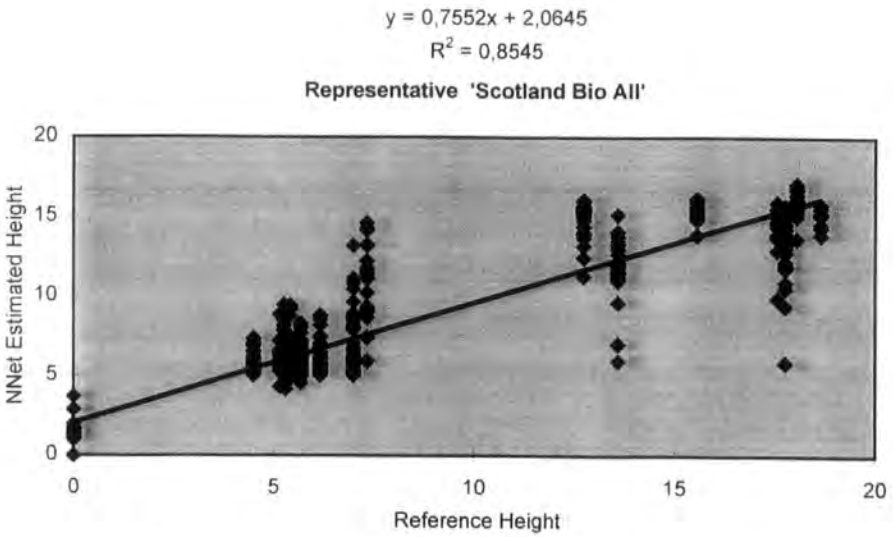


Figure 8-1. Neural network predictions of Height for the representative testing file of the 'Scotland bio all' data set and the continuous target type

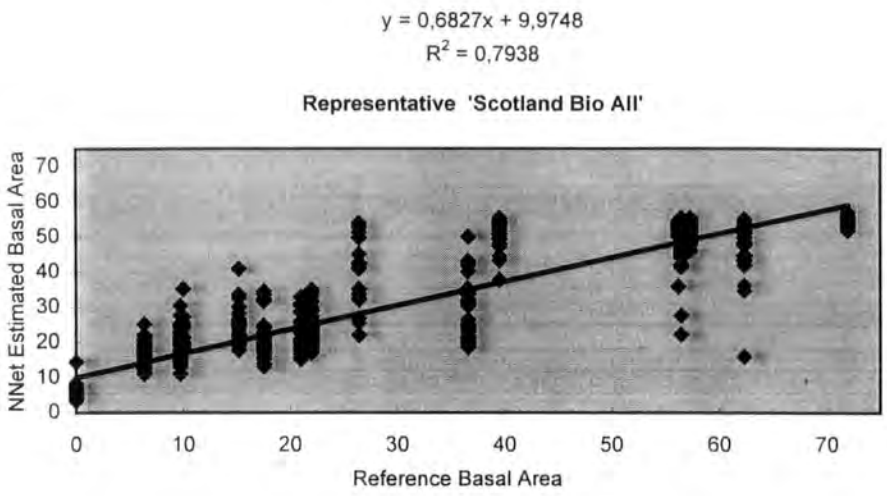


Figure 8-2. Neural network predictions of Basal Area for the representative testing file of the 'Scotland bio all' data set and the continuous target type

The graphs show that the neural network estimates appear to be strongly correlated with reference data for both Basal Area and Height values. Correlation coefficients show that there is a strong positive correlation. The regression explains most of the variance in the graph, as shown by the high R^2 values are high even though there seem to be a number of outliers.

Figure 8-3 and figure 8-4 show the neural network results for the **representative** files from the 'Scotland bio box' data set. The correlation coefficients are: for the Height, $R = +0.93$, $R^2=0.87$; for the Basal Area, $R = +0.91$, $R^2= 0.82$. The correlation coefficients

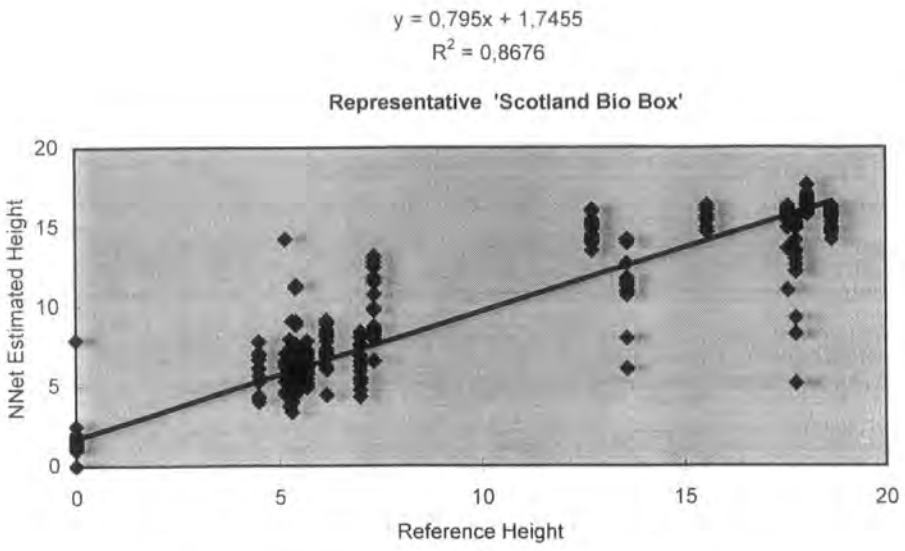


Figure 8-3. Neural network predictions of Height for the representative testing file of the 'Scotland bio box' data set and the continuous target type

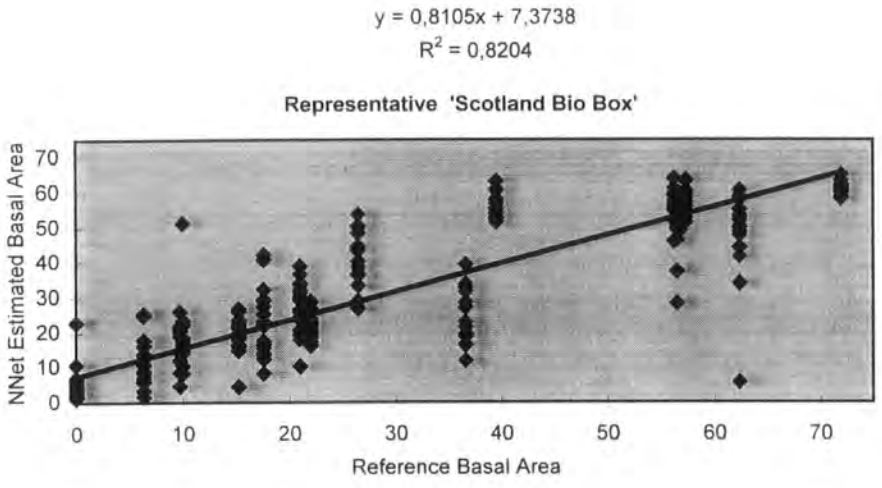


Figure 8-4. Neural network predictions of Basal Area for the representative testing file of the 'Scotland bio box' data set and the continuous target type

are therefore slightly higher for this data set than they are for the previous data set, even though there are still some outliers. Outliers cannot be identified spatially because the locational information of pixels was lost at the data set creation stage.

The results strongly suggest that the neural network is capable of predicting mixed pixel values when it has been trained with a data file containing mixed pixels which is representative of the testing file. The scatter in the results is thought to be partly due to the network's sensitivity to ground data information since it is smaller for the 'Scotland bio box' data set compared to the 'Scotland bio all' data set.

Second and third order polynomials were also fitted to the data to test whether they would explain any more of the variance; in other words, whether the R^2 value increased. It has been suggested that the sigmoid activation value causes the network to bias towards extreme values (Foody, 1996a; Warner and Shank, 1997). The experiments here does not confirm this since third order polynomials did not produce higher R^2 values. The fact that the relationship between Landsat TM channel values and Height and Basal Area decreases as canopy closure occurs (Peterson et al., 1996) could suggest a second order polynomial. There was no evidence of this either however, as R^2 values were lower than for the linear regression.

8.1.3 Results and analysis for the end-member data sets

Figure 8-5 and figure 8-6 present the results for the neural network trained on the **end-member** training files of the 'Scotland bio all' data set. The training file contains only *pure* pixels of **Grass** and **Trees** from closed canopy polygons. The testing file contains only mixed pixels. For this reason, the full range of variable measurements is not present in the testing file. The graphs show that there is no relationship between neural network output values and reference Basal Area and Height measurements for this experiment. Correlation coefficients have fallen to $R = 0.11$ and $R^2 = 0.01$ for Height and $R = 0.11$ and $R^2 = 0.01$ for Basal Area. The neural network therefore does not seem able to predict Basal Area or Height of mixed pixels if it is trained only on *pure* pixels.

Surprisingly, although the network was trained with *pure* **Grass** pixels as well as *pure* **Trees** pixels, when presented with mixed pixels, the network assigns all the mixed pixels high Height and Basal Area values, as if they were *pure* **Trees** pixels; it does not assign any of the pixels to **Grass**, that is zero Height and Basal Area values. This effect cannot be attributed to a higher frequency of closed canopy **Trees** pixels, since *pure* **Grass** pixels constitute 1:3 of the training set. Therefore, this may indicate either that the network is biased towards positive values of a target or that the background **Grass** signal must reach a threshold before the pixel will be assigned to **Grass** in preference to **Trees**. This possibility was also raised in chapter VI, section 6.4.2, when **Medium Density Trees** did not register high output values in the background node.

Figure 8-7 and figure 8-8 present the results for the **end-member** data files from the 'Scotland bio box' data set. The results corroborate the previous findings that there is no relationship between the neural network output values and expected Height and Basal Area measurements. Correlation coefficient have dropped to $R = 0.28$ and $R^2 = 0.08$ for the Height and $R = 0.08$ and $R^2 = 0.01$ for the Basal Area.

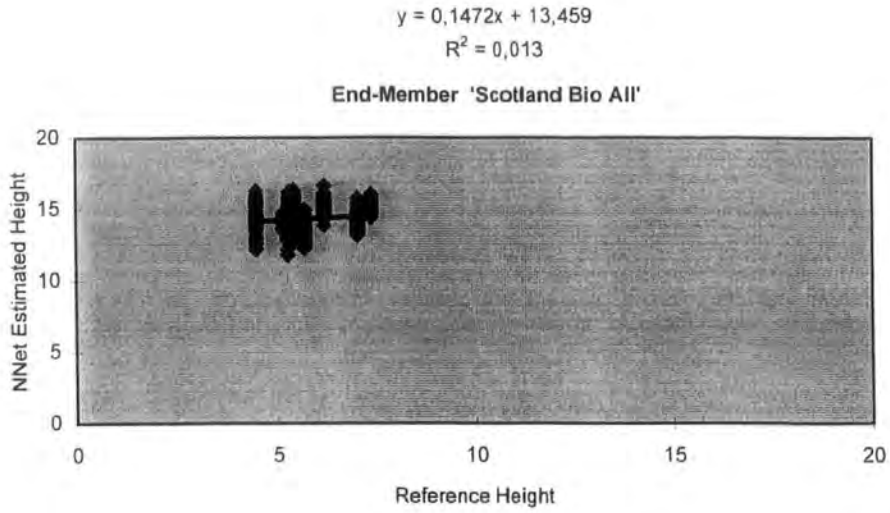


Figure 8-5. Neural network predictions of Height for the end-member testing file of the 'Scotland bio all' data set using a continuous target type

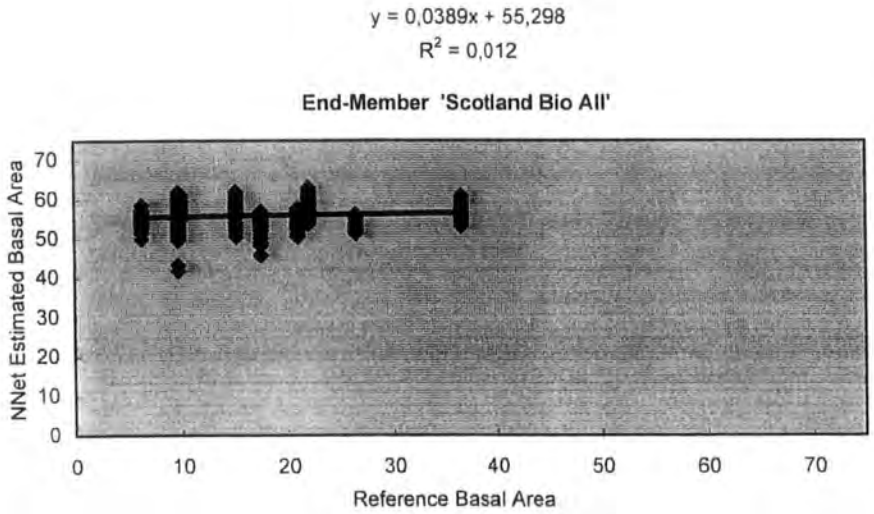


Figure 8-6. Neural network predictions of Basal Area for the end-member testing file of the 'Scotland bio all' data set using a continuous target type

8.1.4 Summary

The results from section 8.1.2 indicate that the neural network can predict mixed pixel continuous variables such as Height and Basal Area measurements with a relatively good level of accuracy according to the graphs and the correlation coefficient. On the other hand, the network cannot estimate variables if it has only been provided with end-

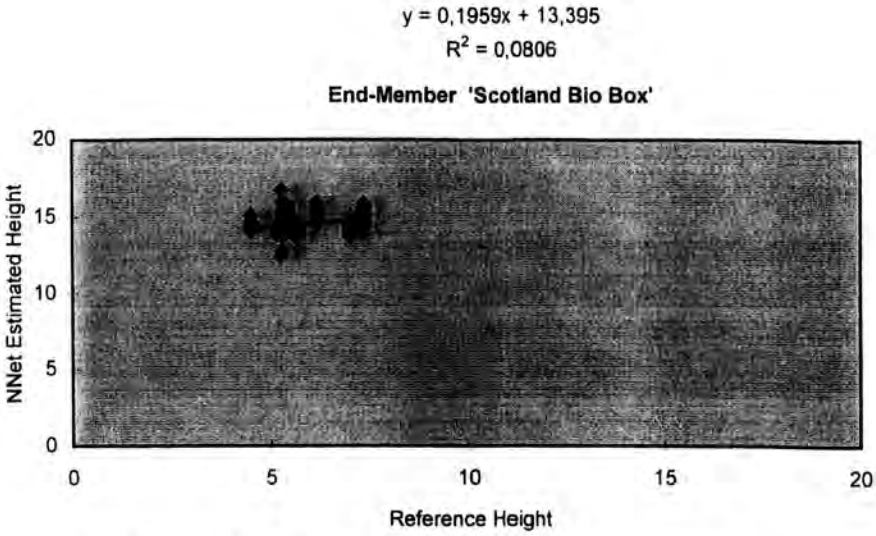


Figure 8-7. Neural network predictions of Height for the end-member testing file of the 'Scotland bio box' data set using a continuous target type

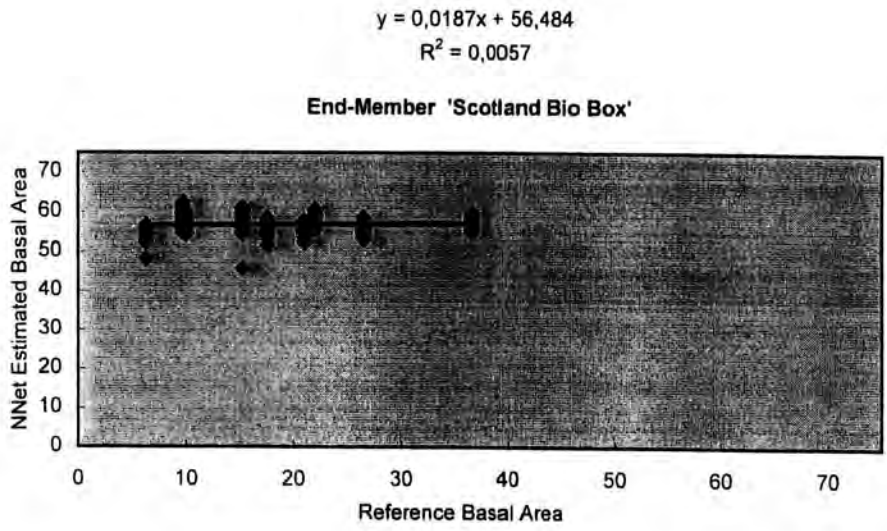


Figure 8-8. Neural network predictions of Basal Area for the end-member testing file of the 'Scotland bio box' data set using a continuous target type

member information as shown in section 8.1.3. The correlation coefficients for each experiment are summarised in table 8-2. As the information in the table shows, correlation values between the predictions of the neural network and actual Height and Basal Area values for the **representative** data set were relatively high whereas they were low for the **end-member** data set. The figures were slightly higher for the 'Scotland bio box' data set than for the 'Scotland bio all' data set except for the approximation of the Basal Area of the **end-member** data set. The network therefore seems to be able to perform a type of regression analysis and interpolate between values if it is trained with a data file that includes mixed pixels but not if it is trained with a data file that only contains *pure* pixels.

	Scotland bio all		Scotland bio box	
Experiment	**R**	**R^2**	**R**	**R^2**
Height 'representative'	0.92	0.85	0.93	0.87
Basal Area 'representative'	0.89	0.79	0.91	0.82
Height 'end-member	0.11	0.01	0.28	0.08
Basal Area 'end-member'	0.11	0.01	0.08	0.01

Table 8-2. Correlation coefficients

Predicting continuous variables is not very common with this type of network. Pierce (1994) used a multi-layer perceptron to predict structural variables from a forest using SAR imagery. The methodology was slightly different in that the input data contained ancillary data and the output values were estimated together, for example Height and Basal Area so that there were two output nodes. The author reports very good predictions but the test sites were the same as the training sites with the only difference being incidence angles and time of day. Gopal and Woodcock (1996) successfully used a multi-layer perceptron to predict change in basal area.

8.2 Correlation between Binary Target and Network Outputs

The experiments were re-run using a binary target similar to the *Bin* targets used in chapter VII. The advantage of these targets is that less precise information is required to create them. Classification was carried out to test whether with this type of target the results obtained in the previous section were confirmed. Since the network appears to extract the relationship between structural variables and Landsat TM values with continuous target values, it should reflect the composition of the pixel when it is trained using a binary target.

8.2.1 Methodology

The same data files as those used in the previous section were used here: the **representative** and **end-member** data files for the 'Scotland bio all' data set and the 'Scotland bio box' data set. The **representative** data sets contain both *pure* and mixed pixels. The **end-member** data sets contain only *pure* pixels in the training file and mixed pixels in the testing file. Neural networks were used with parameters set as shown in table 8-1, except for the output layer of the network which was set to two output nodes instead of one.

The target vector is formed of two elements. The left-most node corresponds to the **Grass** class, the right-most node represents the **Trees** class. Pixels of *pure* **Grass** were given a target of 1 for the **Grass** node and -1 for the **Trees** node. Pixels of *pure* **Trees** were given a target of -1 for the **Grass** node and 1 for the **Trees** node. Pixels of mixed **Grass** and **Trees** were given target values of 0 for each node. This information is summarised in table 8-3.

Description of Pixel	Target
Pure pixel containing only **Grass**	1 -1
Pure pixel containing only **Trees**	-1 1
Mixed pixel containing **Grass** and **Trees**	0 0

Table 8-3. Target for each pixel.

The results from the neural network are displayed using the type of graph used in chapter VI, section 6.4 The vertical axis shows the neural network output values. The expected class is displayed on the x-axis. Class 1 is the **Grass** class, class 2 is the **Mixed** pixels class, class 3 is the **Trees** class. If the network is capable of identifying mixed pixels and their components, then *pure* **Grass** pixels in the testing file should be represented at the output node level by strengths close to +1 and -1 respectively for nodes 1 and 2. *Pure* **Trees** pixels should be represented by strengths close to -1 and 1. **Mixed** pixels' node values should lie about 0 for both nodes.

8.2.2 Results and analysis for the representative data sets

Figure 8-9 shows the response of the network nodes for each class in the **representative** testing file of the 'Scotland bio all' data set (a) and the 'Scotland bio box' data set (b). The response for the **Grass** pixels is close to +1 for the **Grass** node and -1 for the **Trees** node, as expected. The response for the **Trees** class, class 3, is close to -1 for the **Grass** node and +1 for the **Trees** node although there is some spread in the values. There is a large spread of values for the mixed pixels, but the **Grass** node and the **Trees** node do register strengths of 0. The large spread may be due to the fact that the mixed pixels contain a range of mixture proportions. Thus, a pixel cover 75% by **Trees** and 25% by **Grass** would be expected to have a different response than a pixel covered 50% by **Trees** and 50% by **Grass**.

The results for the 'Scotland bio box' data set confirm those for the 'Scotland bio all' data set. However, the spread in values, particularly for the mixed pixels, class 2, has decreased and node strengths are closer to their expected values. This suggests that some

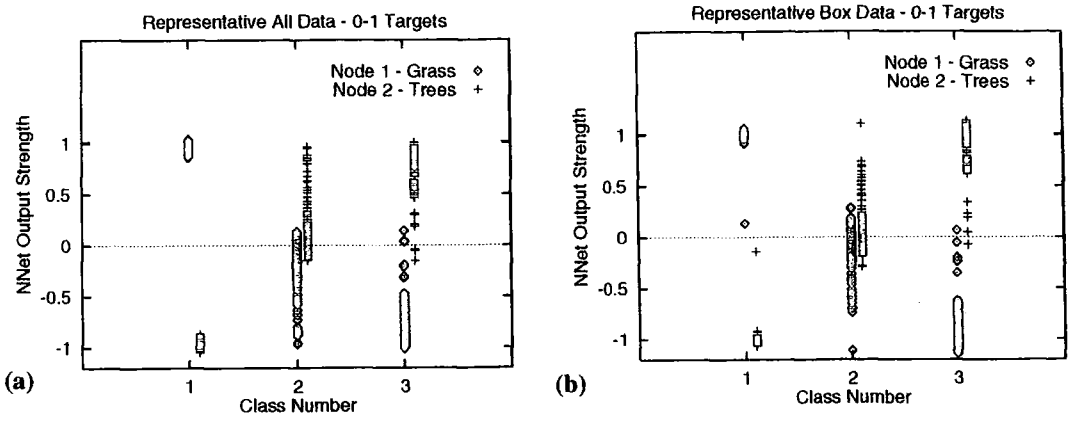


Figure 8-9. Response of the network in each node for pixels from the 'representative' testing file of the 'Scotland bio all' data set (a) and the 'Scotland bio box' data set (b)

of the pixels in the 'Scotland bio all' data set may not actually be well represented by the Height and Basal Area measurements of the single plot. For example, some of the pixels from the polygons with older trees may actually have complete canopy closure, even though the area within which the plot was measured may not.

8.2.3 Results and analysis for the end-member data sets

The same methodology was used for the **end-member** data sets. The results are shown in figure 8-10. The testing set only contains mixed pixels and therefore they are all plotted at the 0 value. The results show that all the pixels are classified as **Trees.** This confirms the results from section 8.1.3 that the neural network cannot use the *pure* signals it has been trained on to identify the mixed pixels by calculating an in-between target value. Instead, all the pixels are classified as -1 for the **Grass** node and +1 for the **Trees** node, indicating **Tree** class. As before, although the network has been trained with some *pure* **Grass** pixels, none of the mixed pixels in the testing file are classified as such.

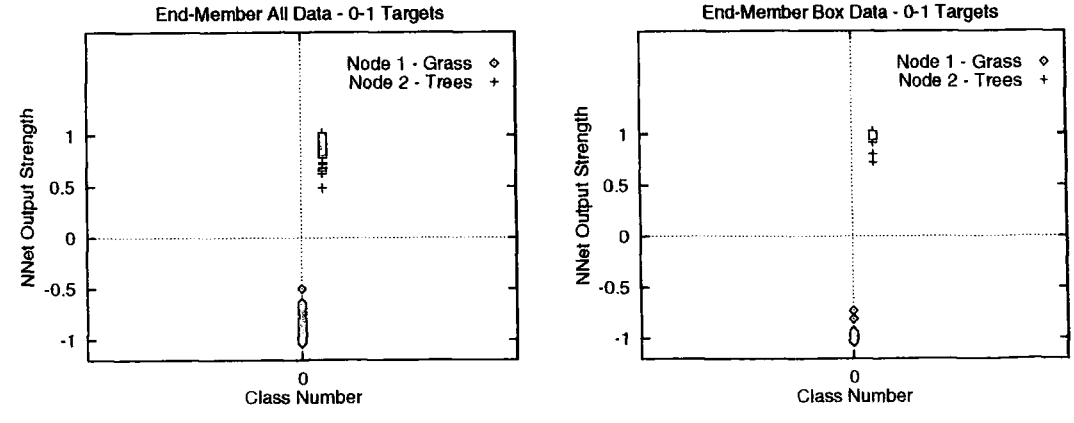


Figure 8-10. Response of the network in each node for pixels from the end-member data set

8.2.4 Summary

The experiments described in this section confirm the results from the experiments in the previous section, section 8.1, for a network trained using a binary target such as those used in traditional *pure* classification problems. If mixed pixels are included in the training data set, then mixed pixel targets are approximated as seen in section 8.2.2 On the other hand, if the neural network is trained only on *pure* classes, mixed pixels targets are not approximated as seen in section 8.2.3. This confirms the suggestion that the neural network must be trained with examples of values that cover the range of data. It does not seem capable of interpolating between *pure* 'end-members'.

8.3 Summary of Chapter VIII

In chapters VI and VII, it was shown that the neural network may respond to the sub-pixel components of a pixel, but a conclusive result as to the degree of accuracy which could be obtained was obscured by the complexity of the data sets. The aim of this chapter was to establish, with a simple, better defined data set, in what way the network responded to the components of a mixed pixel. The data sets consisted of Height and Basal Area measurements used as surrogates for proportions within a mixture.

The results from the experiments described in this chapter have shown that the neural network cannot identify mixed pixels if it is not trained with mixed pixels. This result is true regardless of the target type used to represent reference information. It could be suggested that the neural network has overlearned and therefore cannot predict values in the testing data set. However, the error on the learning data set is still decreasing when the network stops after 600 iterations and it seems unlikely therefore that the network has overlearned.

Surprisingly, in the **end-member** experiments, the network misclassifies all pixels as *pure* **Trees** and none as *pure* **Grass**. It is not clear why that may be but indicates that the response of the network to the presence of **Trees** in a pixel may be non-linear.

The neural network estimations of continuous values have quite a large spread. This suggests that the neural network may require even more precise information than that which it was provided with here. This is confirmed by the reduced variability of the results from the 'Scotland bio box' data set. The neural network estimations of target for the binary target also have quite a large spread. This may be because the network responds to the different proportions within the mixed pixels testing data set.

Chapter IX

SUMMARY AND CONCLUSIONS

Previous research into extracting sub-pixel information suggests that neural networks, and more particularly multi-layer perceptrons, are an attractive alternative to the other main methods that have been investigated such as linear unmixing models, modified maximum likelihood and fuzzy c-means algorithms. Neural networks do not assume that the spectral signature from components of a mixed pixel combine linearly; they do not assume that data is normally distributed, they may not necessarily have to be trained with pure pixels and they are not restricted to any number of classes.

The overall aim of the thesis was to **investigate the relationship between neural network output values and sub-pixel information**. To do so, it was necessary to **address methodological issues concerning the identification of sub-pixel information by the neural network** and to **develop tools for the analysis of *soft* classification results**. This chapter synthesises the results of the experiments that were described in the thesis.

9.1 Preparation

The experiments reported in the literature which suggest a relationship between neural network output values and class proportions within pixels commonly use data sets with simple characteristics. Sometimes the mixed pixels are generated synthetically; often, high resolution data are degraded to produce a coarse resolution image. The number of classes and the number of pixels per class are generally limited. In short, the data sets used in the experiments reported in the literature do not reflect the complexity of typical classification problems. Furthermore, actual pixel proportions are usually estimated from classifications of higher resolution imagery and accurate ground data is not available. Indeed, details of the reference data are not usually forthcoming. Yet without knowledge of the characteristics of the training and testing data sets, the performance of the network cannot be properly appraised. The data sets used in the thesis, described in chapter IV, were created to be representative of the typical landcover classification problem.

Few articles concerned with an application of neural networks describe the network or the methodology for choosing particular parameters of the network in any detail. Nevertheless, an initial study of the performance of the network over a range of values of parameters is essential as it provides the framework within which subsequent experiments can take place. The multi-layer perceptron network used throughout the thesis was described in chapter III. Before carrying out any *soft* classification experiments, the sensitivity of the network to changes to its parameters was evaluated. Results were described in chapter V and show that for appropriate ranges of values, particular choices of parameters do not unduly influence observations.

9.2 Analysis Tools

The overview of the literature summarised in chapter II highlighted a lack of qualitative and quantitative tools for the analysis of *soft* classification results. Several new techniques were developed in the thesis to compare neural network estimated sub-pixel information and actual pixel compositions as provided by reference data and to describe data sets.

9.2.1 *Composition* matrices

Composition matrices, described in chapter IV, section 4.4, characterize the data sets by calculating the frequencies with which classes occur as dominant, secondary, third and so on components within the reference data. From the *Composition* matrices, it is possible to tell how many pixels contain each of the classes and how many times individual classes occur as dominant, secondary and so on components. This provides an *a-priori* overview of the data.

9.2.2 *Correspondence* images

Correspondence images, described in chapter VI, section 6.3, provide a visual account of the results of *soft* classification on the testing data set. Each *pure* class is assigned a colour. For each pixel, compositions are represented by lines coloured proportionally to the area covered by each class within the pixel. Reference data percentages are scaled between zero and one and neural network output values are scaled between zero and one and normalised. *Correspondence* images allow pixel composition according to the reference data to be directly compared with pixel composition according to the neural network output values.

9.2.3 Node response graphs

Node Response Graphs, described in chapter VI, section 6.4, provide a graphical representation of the neural network output values at every node for testing data set pixels grouped according to their dominant class as provided by reference data. The graphs show how classes are represented by network output nodes and interpretations as to the reasons for particular patterns can be made.

9.2.4 *Rank* matrix

The *Rank* matrix, described in chapter VII, section 7.1.2, describes the *soft* classification of the testing data in the form of a matrix. The matrix shows how many classes were correctly identified by the classifier for each position of a mixture, that is dominant, secondary, third and so on. It also shows to what positions mis-identified classes were assigned. Using the *Rank* matrix, it is possible to tell how results were distributed and whether, for example, the classifier tended to assign dominant classes as secondary classes and so on.

9.2.5 *Modified* confusion matrix

The *Modified* confusion matrix, described in chapter VII, section 7.1.3, disregards position information and calculates the frequencies of misclassification between classes. Unlike the traditional confusion matrix, misclassifications are examined at every position. From the *Modified* confusion matrix, it is possible to tell which classes were correctly identified, which were not correctly identified and where classes where mis-allocated to.

9.2.6 Importance of the new analysis tools

Accuracies of classification are a function of class separability, class proportions within pixels and characteristics of the training and testing data sets. Analysis of *soft* classification results should therefore take each of these components into account. The methods that have been developed are best used in conjunction for this reason.

The actual reference data are described by the *Composition* matrices which provides a numerical overview of the characteristics of the data sets. The *Correspondence* images provide the means for an immediate appraisal of the classification results and which classes are confused one with the other. Large data sets may hinder analysis if all the pixels are viewed at the same time but zooming facilities on image processing systems can help. It is suggested that this technique would benefit from additional interactive features such as the possibility of extracting the input spectral signature of a pixel or a group of pixels, for example, or of bringing up the *Composition* matrix for the data set, and so on.

The reasons for which pixel components may have been mis-identified can be partially interpreted from *Node Response Graphs*. The response of a network is usually only plotted for the node with the highest output value whereas the response of the network at every node can provide more insight into the data. For example, *pure* classes that are well defined may tend to have high output strengths in the correct node and low output strengths in the others. Classes that contain mixed pixels may tend to have high output strengths in more than one node. By plotting the output values at every node in *Node Response Graphs*, the behaviour of the network in relation to each group of pixels becomes apparent. In the thesis, pixels were grouped according to their dominant class but any other grouping is possible, for example, by polygon number or by secondary class and so on, although the graphs can get confusing when large numbers of classes are used.

The qualitative information that is provided in the graphs and images is complemented by quantitative methods which provide a numerical basis on which to compare results. The rare numerical methods in the literature are not designed to compare all positions for all pixels. The *Rank* matrix provides detailed information about the distribution of classes for every position and every pixel. This method is only concerned with the relative proportions of classes, not with the classes themselves. The *Modified* misclassification matrices complement the *Rank* matrices by providing the class by class breakdown of results. Although the matrices have been shown to provide useful information concerning the *soft* classification results, they would benefit from a study of the full impact of some of their limitations, described in chapter VII, section 7.1.4.

In summary, the combination of qualitative and quantitative techniques developed in the thesis can provide a better insight into *soft* classification results than was previously available.

9.3 Neural Network Outputs and Sub-Pixel Information

The aims of the experiments described in chapter VI, VII and VIII were to determine the strength of the relationship between neural network output values and class proportions within pixels and to address some methodological issues that had been mostly neglected until now.

9.3.1 Relationship between network outputs and sub-pixel information

In chapter VI, it was shown that the node outputs for network trained and tested with a *pure* data set using a conventional *pure* target were different for correctly classified pixels and incorrectly classified pixels. Hypothesising that incorrectly classified pixels

may in fact have been mixed, the results indicated that the network may have the potential to identify sub-pixel information.

Using a network trained with a data set containing mixed pixels and a conventional *Pure* target, fraction images were created from the neural network outputs on the basis that they approximate *a-posteriori* probabilities and may therefore reflect pixel compositions. The images showed that the dominant classes were generally situated where they were expected spatially. However, most pixels seemed to register a signal for most classes. This was confirmed when *Correspondence* images of the same results were created. Although the dominant class seemed to be correctly identified for most pixels, which was confirmed by an accuracy of ~70%, all the output nodes produced a signal and secondary classes did not appear to be well identified. Using a simpler data set with less classes and fewer mixture types showed that secondary components may be identified for at least some pixels. *Node Response Graphs* for the same data set suggested that classes that were mixed or spectrally similar may produce high output node values in the appropriate classes and low output node values elsewhere. This would suggest that the correct components were being identified.

The qualitative analysis afforded by the *Correspondence* images and *Node Response Graphs* could not provide the level of detail that would allow a conclusive answer regarding the relationship between neural network output values and pixel compositions. Furthermore, it was suggested that the type of target used, which only considered the dominant class of pixels, may be responsible for the apparent lack of a relationship. Quantitative analysis tools were therefore applied to test whether the target type had an influence on classification accuracy and whether there was a relationship between neural network output values and class proportions.

9.3.2 Influence of target types

In chapter VII, the influence of target types was investigated by comparing the *soft* classification outputs for each target type using *Rank* and *Modified* misclassification matrices. The results showed that the neural network appears to be able to identify the component classes of mixed pixels but not their relative proportions. Classes were allocated to the first four positions within a network with a high level of accuracy, but individual position accuracies were low. Target types have a strong influence on the outcome of a classification. Providing information in the target about the composition of a pixel is helpful to the network. However, if the information is too detailed, the classification accuracy can decrease. Target types must therefore be defined with

particular regards to the actual distribution of mixture types. Experiments with a simpler data set suggested that the composition of the data may cause class components to seemingly be identified when it fact it is the nature of the mixing which means that the appropriate secondary classes are highlighted.

9.3.3 Influence of the composition of the data sets

In chapter VI, a network was tested that had been trained only with *pure* classes. The test pixels on the other hand were mixture of some of the *pure* classes. *Node Response Graphs* showed that the appropriate nodes were registering high output values in some cases. This suggested that the network may be able to identify components of a mixture if it is trained only on *pure* classes. This hypothesis was tested in chapter VIII, using a data set with only two *pure* classes and very accurate surrogate measurements for class proportions. The results suggested that the network could not identify the components of a mixture or their proportions if it was only trained on *pure* pixels. On the other hand, a strong relationship between network output values and pixel compositions was obtained if the network had been trained with mixed pixels. Therefore, the network functions as a pattern matching algorithm and not as an unmixing model.

9.3.4 Importance of the results

Experiments in the literature have tended to suggest that the network can not only identify components of mixed pixels, it can also approximate their proportions. However, most experiments use very few classes and few pixels per class. In this thesis, complex and large data sets were used. The results from the experiments suggest that classification accuracy is a function of class separability, proportions within pixels and composition of data sets. The complex inter-relationship between these factors decreases the accuracy with which the network can identify sub-pixel information. It would seem that the network cannot be used on a operational level to identify sub-pixel proportions although it may be able to identify the sub-pixel classes that are present within a pixel.

9.4 Future Directions

The basis of methods which do not use linear or non-linear mixture models is that *a-posteriori* probabilities reflect class proportions. Although it has been shown emperically that this may be the case, it would be useful to understand the exact mathematical reasons for which there may be a relationship between the two. The accuracy of classification of a data set containing mixed pixels was demonstrated to be a

function of class separability, proportions and composition of data sets. It would be interesting to obtain a complete understanding of how these factors inter-relate and their relative importance. The accuracy of the reference data was shown to be a contributing factor to possible misclassifications. The use of reference data sets that were extremely precise and accurate so that exact pixel compositions were known would provide a strong foundation with which to continue the research into extracting sub-pixel information.

References

Adams, J. B., Smith, M. O. & Johnson, P. E. (1986), Spectral mixture modeling: a new analysis of rock and soil types at the Viking Lander 1 Site, *Journal of Geophysical Research* 91(b8), 8098-8112.

Ahern, F. J., Erdle, T., Maclean, D. A. & Kneppeck, I. D. (1991), A quantitative relationship between forest growth rates and Thematic Mapper reflectance measurements, *International Journal of Remote Sensing* 12(3), 387-400.

Aleksander, I. & Morton, H. (1992), An Introduction to Neural Computing, Chapman and Hall, London.

Alimohammadi, A. (1994), Paired classification of digital images using probabilistic measures from maximum-likelihood classifier, in *7th Australasian Remote Sensing Conference Proceedings: Poster and Post Deadline*, Melbourne, 3-8.

Amari, S.-I. (1993), Mathematical methods of neurocomputing, in O. E. Barndorff-Nielsen, J. L. Jensen & W. S. Kendall, eds., Networks and Chaos: Statistical and Probabilistic Aspects, Vol. 50 of Monographs on Statistics and Applied Probability, Chapman & Hall, London, UK, 1-39.

Arai, K. (1992), A supervised Thematic Mapper classification with a purification of training samples, *International Journal of Remote Sensing* 13(11), 2039-2049.

Asner, G. P., Wessman, C. A. & Privette, J. L. (1997), Unmixing the directional reflectances of AVHRR subpixel landcovers, *IEEE Transactions on Geoscience and Remote Sensing* 35(4), 868-878.

Atkinson, P. M., Cutler M.E.J. & Lewis, H. (1997), Mapping sub-pixel proportional land cover with AVHRR imagery, *International Journal of Remote Sensing* 18(4), 917-935.

Bartl, R. & Pinz, A. (1992). Information fusion in remote sensing land use classification, in *Proceedings of the IAPR TC7 Workshop*, 9-17.

Bastin, L. (1997), Comparison of fuzzy c-means classification, linear mixture modelling and MLC probabilities as tools for unmixing coarse pixels, ***International Journal of Remote Sensing*** 18(17), 3629-3648.

Bateson, A. & Curtiss, B. (1996), A method for manual endmember selection and spectral unmixing, *Remote Sensing of the Environment* 55, 229-243.

Baum, E. D. & Haussler, D. (1989), What size net gives valid generalization?, in D. S. Touretzky, ed., Advances in Neural Information Processing Systems, Morgan Kauffman, San Mateo Ca, USA, 81-90.

Benediktsson, J. A. & Sveinsson, J. R. (1997), Feature extraction for neural network classifiers, in I. Kanellopoulos, G. G. Wilkinson, F. Roli & J. Austin, eds.., Neurocomputation in Remote Sensing Data Analysis, Springer, Berlin - Heidelberg, 97-105.

Benediktsson, J. A., Swain, P. H. & Ersoy, O. K. (1990), Neural network approaches versus statistical methods in classification of multisource remote sensing data, *IEEE Transactions on Geoscience and Remote Sensing* 28(4), 540-552.

Bernard, A. C. & Donoghue, D. (1996), Classifying forest pixels using neural networks, in *Proceedings of the 22nd annual conference of the Remote Sensing Society*, Remote Sensing Society, UK, 411-418.

Bernard, A. C., Kanellopoulos, I. & Wilkinson, G. G. (1995), Neural network classification of mixtures, in E. Binaghi, P. A. Brivio & A. Rampini, eds., Soft Computing in Remote Sensing Data Analysis, Milano, Italy, Vol. 1 of Series in Remote Sensing, Consiglio Nazionale Delle Ricerche, Italy, World Scientific, Singapore, 53-58.

Bernard, A. C., Wilkinson, G. G. & Kanellopoulos, I. (1997), Training strategies for neural network soft classification of remotely-sensed imagery, *International Journal of Remote Sensing* 18(8), 1851-1856.

Bezdek, J. C. (1984), FCM. The fuzzy c-means clustering algorithm, *Computers and Geosciences* 10, 191-203.

Bischof, H., Schneider, W. & Pinz, A. J. (1992), Multispectral classification of Landsat images using neural networks, *IEEE Transactions on Geoscience and Remote Sensing* 30(3), 482-490.

Bishop, C. M. (1995), Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK.

Borel, C. C. & Gerstl, S. A. W. (1994), Nonlinear spectral mixing models for vegetative and soil surfaces, *Remote Sensing of the Environment* 47, 403-416.

Bosdogianni, P., Petrou, M. & Kittler, J. (1997), Mixture models with higher order moments, *IEEE Transactions on Geoscience and Remote Sensing* 35(2), 341-353.

Bottou, L. (1991), Une Approche Théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole, PhD thesis, Université de Paris-Sud.

Brockhaus, J. A. & Khorram, S. (1992), A comparison of SPOT and Landsat-TM data for use in conducting inventories of forest resources, *International Journal of Remote Sensing* 13(16), 3035-3043.

Bryant, R. G. (1996), Validated linear mixture modelling of Landsat TM data for mapping

evaporite minerals on a playa surface: methods and applications, *International Journal of Remote Sensing* 17(2), 315-330.

Butera, M. K. (1986), A correlation and regression analysis of percent canopy closure versus TMS spectral response for selected forest sites in the San Juan National Forest, Colorado, *IEEE Transactions on Geoscience and Remote Sensing* 24(1), 122-129.

Campbell, J. B. (1981), Spatial correlation effects upon accuracy of supervised classification of land cover, *Photogrammetric Engineering and Remote Sensing* 74(3), 355-363.

Campbell, J. B. (1996), Introduction to Remote Sensing, 2nd edn, The Guilford Press, New York.

Campbell, J. G. & Hashim, A. A. (1992), Fuzzy sets, pattern recognition, linear estimation and neural networks - a unification of the theory with relevance to remote sensing, in R. A. Cracknell & R. A. Vaughan, eds., *Proceedings of the 18th Annual Conference of the UK Remote Sensing Society*, Dundee, Scotland, Remote Sensing Society, UK, 508-517.

Cannon, R. L., Dave, J. V. and Bezdek, J. C. (1986a), Efficient implementation of the fuzzy c-means clustering algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (2), 248-255.

Cannon, R. L., Dave, J. V., Bezdek, J. C. & Tivedi, M. M. (1986b), Segmentation of a Thematic Mapper image using the fuzzy c-means clustering algorithm, *IEEE Transactions on Geoscience and Remote Sensing* 24(3), 400-408.

Canters, F. (1997), Evaluating the uncertainty of area estimates derived from fuzzy land cover classification, *Photogrammetric Engineering and Remote Sensing* 63(4), 403-414.

Cheeseman, P. (1985), In defense of probability, in *Proceedings of the 9th IJCAI*, 18-23 August 1985, 1002-1009.

Chhikara, R. S. (1984), Effect of mixed (boundary) pixels on crop proportion estimation, *Remote Sensing of the Environment* 14, 207-218.

Cohen, W. B. & Spies, T. A. (1992), Estimating structural attributes of Douglas-Fir/ Western Hemlock forest stands from Landsat and SPOT imagery, *Remote Sensing of the Environment* 41, 1-17.

Congalton, R. G. (1991), A review of assessing the accuracy of classifications of remotely sensed data, *Remote Sensing of the Environment* 37, 35-46.

Cortez, L., Durão, F. and Ramos, V. (1997), Testing some connectionist approaches for

thematic mapping of rural areas, in I. Kanellopoulos, G. G. Wilkinson, F. Roli & J. Austin, eds.., Neurocomputation in Remote Sensing Data Analysis, Springer, Berlin - Heidelberg, 142-150.

Cracknell, A. P. (1998), Synergy in remote sensing - what's in a pixel?, *International Journal of Remote Sensing* 19(11), 2025-2047.

Craig, M. D. (1994), Minimum-volume transforms for remotely sensed data, *IEEE Transactions on Geoscience and Remote Sensing* 32(3), 542-552.

Cross, A. M., Settle, J. J., Drake, N. A. & Paivinen, R. T. M. (1991), Subpixel measurement of tropical forest cover using AVHRR data, *International Journal of Remote Sensing* 12(5), 1119-1129.

Czaplewski, R. L. (1992), Misclassification bias in areal estimates, *Photogrammetric Engineering and Remote Sensing* 58(2), 189-192.

Danson, F. M. & Curran, P. J. (1993), Factors affecting the remotely sensed response of coniferous forest plantations, *Remote Sensing of the Environment* 43, 55-65.

Dayhoff, J. E. (1990), Neural Network Architectures: an Introduction, Van Nostrand Reinhold, New York.

Detchmendy, D. M. & Pace, W. H. (1972), A model for spectral signature variability for mixtures, in *Earth Resources Observations and Information Analysis Conference*, Vol. 1, Tullahoma, Tennessee, US, 596- 620.

van Deusen, P. C. (1995), Modified highest confidence first classification, *Photogrammetric Engineering and Remote Sensing* 61(4), 419-425.

Donoghue, D. N. M. (1994), Estimation of forest canopy densities using Landsat TM data with a linear mixture model, in P. J. Kennedy and M. Karteris, eds., *Proceedings of the International Workshop on Satellite Technology and GIS for Mediterranean Forest Mapping and Fire Mangement*, EUR 15861-EN, 397-406.

Donoghue, D., Reid-Thomas, D. C. & Zong, Y. (1995), Mapping and monitoring the intertidal zone of the east coast of England using remote sensing techniques and a coastal monitoring GIS, *Marine Technology Society Journal* 28(2), 19-29.

Downey, I. D., Power, C. H., Kanellopoulos, I. & Wilkinson, G. G. (1992), A performance comparison of Landsat TM land cover classification based on neural network techniques and traditional maximum likelihood and minimum distance algorithms, in R. A. Cracknell & R. A. Vaughan, eds., *Proceedings of the 18th Annual Conference of the UK Remote Sensing Society*, Dundee, Scotland, Remote Sensing Society, London, UK, 518-528.

Dreyer, P. (1993), Classification of land cover using optimized neural nets on SPOT data,

Photogrammetric Engineering and Remote Sensing 59(5), 617-621.

Fierens, F., Wilkinson, G. G. & Kanellopoulos, I. (1994), Studying the behaviour of neural and statistical classifiers by interaction in feature space, in J. Desachy, ed., Image and Signal Processing for Remote Sensing, Proceedings SPIE 2315, 483-493.

Fisher, P. F. (1994a), Hearing the reliability in classified remotely sensed images, *Cartography and Geographic Information Systems* 21(1), 31- 36.

Fisher, P. F. (1994b), Visualisation of the reliability in classified remotely sensed images, *Photogrammetric Engineering and Remote Sensing* 60(7), 905-910.

Fisher, P. F. (1997), The pixel: a snare and a delusion, *International Journal of Remote Sensing* 18(), 679-685.

Fisher, P. F. (1996), Boolean and fuzzy regions, in Burrough, P.A. & Frank, A.U. eds. (1996), Geographic Objects with Indeterminate Boundaries, Taylor and Francis, London, UK, 87-93.

Fisher, P. F. & Pathirana, S. (1990), The evaluation of fuzzy membership of land cover classes in the suburban zone, *Remote Sensing of the Environment* 34, 121-132.

Flexer, A. (1995), Statistical evaluation of neural network experiments: minimum requirements and current practice, Technical Report oefai-tr-95-16, The Austrian Research Institute for Artificial Intelligence, Vienna, Austria.

Fogelman-Soulie, F. (1991), Neural network architectures and algorithms: a perspective, in T. Kohonen, K. Makisara, O. Simula & J. Kangas, eds., *Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91)*, Espoo, Finland, Vol. 1, North-Holland.

Foody, G. M. (1992), A fuzzy sets approach to the representation of vegetation continua from remotely sensed data: an example from lowland heath, *Photogrammetric Engineering and Remote Sensing* 58(2), 221- 225.

Foody, G. M. (1994), Ordinal-level classification of sub-pixel tropical forest cover', *Photogrammetric Engineering and Remote Sensing* 60(1), 61-65.

Foody, G. M. (1996a), Approaches for the production and evaluation of fuzzy land cover classifications from remotely-sensed data, *International Journal of Remote Sensing* 17(7), 1317-1340.

Foody, G. M. (1996b), Relating the land-cover composition of mixed pixels to artificial neural network classification output, *Photogrammetric Engineering and Remote Sensing* 62(5), 491-499.

Foody G. M. & Arora, M. K. (1996), Fuzzy thematic mapping: incorporating mixed pixels

in the training, allocation and testing stages of supervised image classification, in E. Binaghi, P. A. Brivio & A. Rampini, eds., Soft Computing in Remote Sensing Data Analysis, Milano, Italy, Vol. 1 of Series in Remote Sensing, Consiglio Nazionale Delle Ricerche, Italy, World Scientific, Singapore, 43-52.

Foody, G. M. & Arora, M. K. (1997), An evaluation of the factors affecting neural network classification accuracy, *International Journal of Remote Sensing,* 18(4), 799-810.

Foody, G. M. & Cox, D. P. (1994), Sub-pixel land cover composition estimation using a linear mixture model and fuzzy membership functions, *International Journal of Remote Sensing* 15(3), 619-631.

Foody, G. M., Campbell, N. A., Trodd, N. M. & Wood, T. F. (1992), Derivation and applications of probabilistic measures of class membership from maximum-likelihood classification, *Photogrammetric Engineering and Remote Sensing* 58(9), 1335-1341.

Foody, G. M., McCulloch, M. B. & Yates, W. B. (1995a), Classification of remotely sensed data: issues related to training data characteristics, *Photogrammetric Engineering and Remote Sensing* 61(4), 391-402.

Foody, G. M., McCulloch, M. B. & Yates, W. B. (1995b), The effect of training set size and composition on neural network classification, *International Journal of Remote Sensing* 16(9), 1707-1724.

Foody, G. M., Lucas, R. M., Curran, P. & Honzak, M. (1997), Non-linear mixture modelling without end-members using an artificial neural network, *International Journal of Remote Sensing* 18(4), 937-953.

Foschi, P. G. & Smith, D. K. (1997), Detecting sub-pixel woody vegetation in digital imagery using two artificial intelligence approaches, *Photogrammetric Engineering and Remote Sensing* 63(5), 493-500.

Franklin, J. (1986), Thematic mapper analysis of coniferous forest structure and composition, *International Journal of Remote Sensing* (7), 1287- 1301.

Franklin, S. E. (1994), Discrimination of subalpine forest species and canopy density using digital CASI, SPOT PLA, and Landsat TM data, *Photogrammetric Engineering and Remote Sensing* 60(10), 1233-1241.

Franklin, J., Logan, T. L., Woodcock, C. E. & Strahler, A. H. (1986), Coniferous forest classification and inventory using Landsat and digital terrain data, *IEEE Transactions on Geoscience and Remote Sensing* 24(1), 139-148.

klein Gebbinck, M. & Schouten, T. E. (in print), Crop area estimation by data driven

decomposition of mixed pixels in Landsat TM images, *International Journal of Remote Sensing*.

Gemmell, F. M. & Goodenough, D. G. (1992), Estimating timber volume from TM data: the importance of scale and accuracy of forest cover data, in R. A. Cracknell & R. A. Vaughan, eds, *Proceedings of the 18th Annual Conference of the UK Remote Sensing Society*, Dundee, Scotland, Remote Sensing Society, 297-306.

Gong, P., Miller, J. R. & Spanner, M. (1994), Forest canopy closure from classification and spectral unmixing of scene components - multisensor evaluation of an open canopy, *IEEE Transactions on Geoscience and Remote Sensing* 32(5), 1067-1080.

Gopal, S. & Woodcock, C. (1994), Theory and methods for accuracy assessment of thematic maps using fuzzy sets, *Photogrammetric Engineering and Remote Sensing* 60(2), 181-188.

Hallum, C. R. (1972), On a model for optimal proportions estimates for category mixtures, in *Proceedings of the 8th International Symposium on Remote Sensing of the Environment*, Ann Arbor, Michigan, US, 951- 958.

Heerman, P. D. & Khazenie, N. (1992), Classification of multispectral remote sensing data using a back-propagation neural network, *IEEE Transactions on Geoscience and Remote Sensing* 30(1), 81-88.

Hepner, G. F., Logan, T., Ritter, N. & Bryant, N. (1990), Artificial neural network classification using a minimal training set: comparison to conventional supervised classification, *Photogrammetric Engineering and Remote Sensing* 56(4), 469-473.

Hernandez, R., Varfis, A., Kanellopoulos, I. & Wilkinson, G. G. (1992), Development of MLP/LVQ hybrid networks for classification of remotely-sensed satellite images, in I. Aleksander & J. Taylor, eds., *Proceedings of the 1992 International Conference on Artificial Neural Networks (ICANN-92)*, Brighton, U.K., Vol. 2, Elsevier Science Publications (North-Holland), 1193-1196.

Holben, B. N. & Shimabukuro, Y. E. (1993), Linear mixing model applied to coarse spatial resolution data from multispectral satellite sensors, *International Journal of Remote Sensing* 14(11), 2231-2240.

Horler, D. N. H. & Ahern, F. J. (1986), Forestry information content of Thematic Mapper data, *International Journal of Remote Sensing* 7(3), 405-428.

Hornik, K., Stinchcombe, M. & White, H. (1989), Multilayer feedforward networks are universal approximators, *Neural Networks* 2, 359-366.

Horwitz, H. M., Nalepka, R. F., Hyde, P. D. & Morgenstern, J. P. (1971), Estimating the proportions of objects within a single resolution element of a multispectral scanner,

Proceedings of the 7th International Symposium on Remote Sensing of the Environment, Ann Arbor, Michigan, US, 1307-1320.

Howald, K. J. (1989), Neural network image classification, *Proceedings ASPRS-ACSM Fall Convention*, Cleveland, 207-215.

Huguenin, R. L., Karaska, M. A. & Blaricom, D. V. & Jensen, J. R. (1997), Subpixel classification of bald cypress and tupelo gum trees in Thematic Mapper imagery, *Photogrammetric Engineering and Remote Sensing* 63(6), 717-725.

Hush, D. R. & Horne, B. G. (1993), Progress is supervised neural networks, *IEEE Signal Processing Magazine* 10, 8-39.

Irons, J. R., Markham, B. L., Nelson, R. F., Toll, D. L., Williams, D. L., Latty, R. S. & Stauffer, M. L. (1985), The effects of spatial resolution on the classification of Thematic Mapper data, *International Journal of Remote Sensing* 6(8), 1385-1403.

Janssen, L. L. F. & van der Wel, F. J. M. (1994), Accuracy assessment of satellite derived land cover data: a review, *Photogrammetric Engineering and Remote Sensing* 60(4), 419-426.

Jasinski, M. F. & Eagleson, P. S. (1990), Estimation of subpixel vegetation cover using red-infrared scattergrams, *IEEE Transactions on Geoscience and Remtoe Sensing* 28(2), 253-267.

Jasinski, M. F. (1996), Estimation of subpixel vegetation density of natural regions using satellite multispectral imagery, *IEEE Transactions on Geoscience and Remote Sensing* 34(3), 804-813.

Jupp, D. L. B & Walker, J. (1997), Detecting structural and growth changes in woodlands and forests: the challenge for remote sensing and the role of geometric optical modeling, in H. L. Gholz, K. Nakane & H. Shimoda, eds., The Use of Remote Sensing in the Modeling of Forest Productivity, Kluwer Academic Publishers, Dordrecht, chapter 4.

Kanellopoulos, I. & Wilkinson, G. G. (1997), Strategies and best practice for neural network image classification, *International Journal of Remote Sensing* 18(4), 711-725.

Kanellopoulos, I., Varfis, A., Wilkinson, G. G. & Megier, J. (1991), Classification of remotely sensed satellite images using multi-layer perceptron networks, in T. Kohonen, K. Makisara, O. Simula & J. Kangas, eds., *Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91)*, Espoo, Finland, Vol. 2, Elsevier Science Publications (North-Holland), 1067-1070.

Kanellopoulos, I., Varfis, A., Wilkinson, G. G. & Megier, J. (1992), Land cover

discrimination in SPOT HRV imagery using an artificial neural network: a 20-class experiment, *International Journal of Remote Sensing* 13(5), 917-924.

Kanellopoulos, I., Wilkinson, G. G. & Megier, J. (1993), Integration of neural network and statistical image classification for land cover mapping, in *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS'93)*, Tokyo, Japan, Vol. II, IEEE Press, 511- 513.

Kanellopoulos, I., Fierens, F. & Wilkinson, G. G. (1994a), Combination of parametric and neural classifiers for analysis of multi-sensor remote sensing imagery, S.-S. Chen, ed., Neural and Stochastic Methods in Image and Signal Processing III, Proceedings SPIE 2304, 218-224.

Kanellopoulos, I., Wilkinson, G. G. & Chiuderi, A. (1994b), Land cover mapping using combined Landsat TM imagery and textural features from ERS-1 Synthetic Aperture Radar Imagery, in J. Desachy, ed., Image and Signal Processing for Remote Sensing, Proceedings SPIE 2315, 332-341.

Kent, J. T. & Mardia, K. V. (1988), Spatial classification using fuzzy membership models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(5), 659-670.

Kerdiles, H. & Grondona, M. (1995), NOAA-AVHRR NDVI decomposition and subpixel classification using linear mixing in the Argentinian Pampa, *International Journal of Remote Sensing* 16(7), 1303-1325.

Kimes, D. S., Nelson, R. F., Manry, M. T. & Fung, A. K. (1998), Attributes of neural networks for extracting continuous vegetation variables from optical and radar measurements, *International Journal of Remote Sensing* 19(14), 2639-2663.

Klir, G. J. & Yuan, B. (1995), Fuzzy Sets and Fuzzy Logic, Prentice Hall, US.

Kohonen, T. (1989), Self-Organization and Associative Memory, 3re ed. Springer-Verlad, Berlin, Germany.

van Kootwijk, E. J., der Voet, H. V. & Berdowski, J. J. M. (1995), Estimation of ground cover composition per pixel after matching image and ground data with subpixel accuracy, *International Journal of Remote Sensing* 16(1), 97-111.

Kosko, B. (1992), Neural Networks and Fuzzy Systems: a Dynamical Systems Approach to Machine Intelligence, Prentice Hall, New Jersey.

Kung, S. Y. (1993), Digital Neural Networks, Prentice Hall Information and System Sciences Series, Prentice Hall, New Jersey,US.

Lagacherie, P., Andrieux, P. & Bouzigues, R. (1996), Fuzzines and uncertainty of soil boundaries: from reality to coding in GIS, in Burrough, P.A. & Frank, A.U. eds., Geographic Objects with Indeterminate Boundaries, Taylor and Francis, London,

UK, 275-285.

LeCunn, Y. (1987), Modèles Connexionnistes de l'Apprentissage, PhD thesis, Université Marie Curie, Paris 6.

Lee, H.-M. & Wang, W.-T. (1994), A neural network architecture classification of fuzzy inputs, *Fuzzy Sets and Systems* 63, 159-173.

Li, X. & Strahler, A. H. (1986), Geometric-optical bidirectional reflectance modeling of a conifer forest canopy, *IEEE Transactions on Geoscience and Remote Sensing* 24(6), 907-919.

Lillesand, T. M. & Kiefer, R. W. (1994), Remote Sensing and Image Interpretation, 3rd edn, John Wiley and Sons Inc, New York.

Lisboa, P. G. J. (1992), Neural Networks: Current Applications, Chapman & Hall, London.

Liu, Z. K. & Xiao, J. Y. (1991), Classification of remotely sensed image data using artificial neural networks, *International Journal of Remote Sensing* 12(11), 2433-2438.

Lowe, D. & Webb, A. R. (1991), Optimized feature extraction and the Bayes decision in feed-forward classifier networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(4), 355-364.

Marsh, S. E., Switzer, P. & Lyon, R. J. P. (1980), Resolving the percentage of component terrains within single resolution elements, *Photogrammetric Engineering and Remote Sensing* 46(8), 1079-1086.

Maselli, F., Conese, C., Filippis, T. D. & Norcini, S. (1995), Estimation of forest parameters through fuzzy classification of TM data., *IEEE Transactions on Geoscience and Remote Sensing* 33(1), 77-84.

Maselli, F., Rodolfi, A. & Conese, C. (1996), Fuzzy classification of spatially degraded Thematic Mapper data for the estimation of sub-pixel components, *International Journal of Remote Sensing* 17(3), 537-551.

Mathieu-Marni, S., Leymarie, P. & Berthod, M. (1996), Removing ambiguities in a multi-spectral image classification, *International Journal of Remote Sensing* 17(8), 1493-1504.

Mayaux, P. & Lambin, E. F. (1995), Estimation of tropical forest area from coarse spatial resolution data: a two-step correction function for proportional errors due to spatial aggregation, *Remote Sensing of the Environment* 53, 1-15.

McCloy, K. R. & Hall, K. A. (1991), Mapping the density of woody vegetative cover using Landsat MSS digital data, *International Journal of Remote Sensing* 12(9),

1877-1885.

Moody, A., Gopal, S. & Strahler, A. H. (1996), Artificial neural network response to mixed pixels in coarse resolution satellite data, *Remote Sensing of the Environment* 58, 329-343.

Oleson, K. W., Sarlin, S., Garrison, J., Smith, S., Privette, J. L. & Emery, W. J. (1995), Unmixing multiple land cover type reflectances from coarse spatial resolution satellite data, *Remote Sensing of the Environment* 54, 98-112.

Osman, H. & Fahmy, M. M. (1994), On the discriminatory power of adaptive feed forward layered networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(8), 837-842.

Paola, J. D. & Schowengerdt, R. A. (1994), Comparison of neural networks to standard techniques for image classification and correlation, *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS'94)*, Pasadena, California, IEEE press, 1404-1406.

Paola, J. D. & Schowengerdt, R. A. (1995a), A detailed comparison of backpropagation neural network and maximum-likelihood classifiers for urban land use classification, *IEEE Transactions on Geoscience and Remote Sensing* 33(4), 981-996.

Paola, J. D. & Schowengerdt, R. A. (1995b), A review and analysis of backpropagation neural networks for classification of remotely sensed multi-spectral imagery, *International Journal of Remote Sensing* 16(16), 3033-3058.

Paola, J. D. & Schowengerdt, R. A. (1997), The effect of neural-network structure on a multispectral land-use/land-cover classification, *Photogrammetric Engineering and Remote Sensing* 63(5), 535-544.

Peterson, D. L., Westman, W. E., Stephenson, N. J., Ambrosia, V. G., Brass, J. A & Spanner, M. A. (1986), Analysing forest structure using TM simulator data, *IEEE Transactions on Geoscience and Remote Sensing* 24(1), 113-120.

Pham, D. T. & Bayro-Corrochano, E. J. (1994), Self-organizing neural network based pattern clustering method with fuzzy outputs, *Pattern Recognition* 27(8), 1103-1110.

Pierce, L. E., Sarabandi, K. & Ulaby, F. T. (1994), Application of an artificial neural network in canopy scattering inversion, *International Journal of Remote Sensing* 15(16), 3263-3270.

Puhr, C. B. (1997), Catchment afforestation, surface water acidification and salmonid population in Galloway, South West Scotland, PhD thesis, University of Durham.

Puri, M. L. & Ralescu, D. (1982), A possibility measure is not a fuzzy measure, *Fuzzy Sets and Systems* 7, 311-313.

Quarmby, N. A., Townshend, J. R. G., Settle, J. J., White, K. H., Milnes, M., Hindle, T. L. & Silleos, N. (1992), Linear mixture modelling applied to AVHRR data for crop area estimation, *International Journal of Remote Sensing* 13(3), 415-425.

Ray, T.W. & Murray, B.C. (1996), Nonlinear spectral mixing in desert vegetation, *Remote Sensing of the Environment* 49, 13-20.

Reid-Thomas, D. C., Donoghue, D. M. & Shennan, I. (1995), Intertidal vegetation mapping using Landsat 5 Thematic Mapper data, in Healy & Doody, eds., Directions in European Coastal Management, Samara Publishing Limited, Cardigan, 213-222.

Richard, M. D. & Lippmann, R. P. (1991), Neural network classifiers estimate Bayesian a-posteriori probabilities, *Neural Computation* 3, 461- 483.

Richards, J. A. (1986), Remote Sensing Digital Image Analysis: An Introduction, Springer-Verlag, Berlin.

Ripley, B. D. (1993), Statistical aspects of neural networks, in O. E. Barndorff-Nielsen, J. L. Jensen & W. S. Kendall, eds., Networks and Chaos: Statistical and Probabilistic Aspects, Vol. 50 of Monographs on Statistics and Applied Probability, Chapman & Hall, London, UK, 40-123.

Rosenfield, G. H. & Fitzpatrick-Lins, K. (1986), A coefficient of agreement as a measure of thematic classification accuracy, *Photogrammetric Engineering and Remote Sensing* 52(2), 223-227.

Rosin, P. L. & Fierens, F. (1995), Improving neural network generalisation, in *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS'95)*, Florence, Italy.

Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E. & Suter, B. W. (1990), The multilayer perceptron as an approximation to a Bayes optimal discriminant function, *IEEE Transactions on Neural Networks* 1(4), 296-298.

Rumelhart, D. E., McClelland, J. L. & the PDP Research Group, eds. (1986a), Parallel Distributed Processing. Explorations in the Micro Structures of Cognition, Vol. 1: Foundations, MIT Press.

Sader, S. A. & Winne, J. C. (1992), RGB-NDVI colour composites for visualising forest change dynamics, *International Journal of Remote Sensing* 13(16), 3055-3067.

Schott, J. R. (1997), Remote Sensing: the Image Chain Approach, Oxford University Press, New York.

Schouten, T. E. & klein Gebbinck, M. S. (1997), A neural network approach to spectral mixture analysis, in I. Kanellopoulos, G. G. Wilkinson, F. Roli & J. Austin, eds., Neurocomputation in Remote Sensing Data Analysis, Springer, Berlin - Heidelberg, 79-85.

Settle, J. & Campbell, N. (1998), On the errors of two estimators of sub-pixel fractional cover when mixing is linear, *IEEE Transactions on Geoscience and Remote Sensing* 36(1), 163-170.

Settle, J. J. & Drake, N. A. (1993), Linear mixing and the estimation of ground cover proportions, *International Journal of Remote Sensing* 14(6), 1159-1177.

Shimabukuro, Y. E. & Smith, J. A. (1991), The least squares mixing models to generate fraction images derived from remote sensing multispectral data, *IEEE Transactions on Geoscience and Remote Sensing* 29(1), 16-20.

Shoemaker, P. A. (1991), A note on least squares learning procedures and classification by neural network models, *IEEE Transactions on Neural Networks* 2(1), 158-160.

Simpson, P. K. (1990), Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations, Neural Networks: Research and Applications, Pergamon Press, New York, US.

Skidmore, A. K., Turner, B. J., Brinkhof, W. & Knowles, E. (1997), Performance of a neural network: mapping forests using GIS and remotely sensed data, *Photogrammetric Engineering and Remote Sensing* 63(5), 501-514.

Smith, M. O., Johnson, P. E. & Adams, J. B. (1985), Quantitative determination of mineral types and abundances from reflectance spectra using principal component analysis, *Journal of Geophysical Research* 90, 792-804.

Smith, M. O., Adams, J. B. & Gillespie, A. R, (1990), Reference endmembers for spectral mixture analysis, *Proceedings of the 5th Australasian Remote Sensing Conference*, Vol 1., 331-340.

Smith, M. O., Adams, J. B. & Sabol, D. E. (1994), Spectral mixture analysis - new strategies for the analysis of multispectral data, in J. Hill & J. Megier, eds., Imaging Spectrometry - a Tool for Environmental Observations, EEC, Netherlands, 125-143.

Sohn, Y. & McCoy, R. M. (1997), Mapping desert shrub rangeland using spectral unmixing and modeling spectral mixtures with TM data, *Photogrammetric Engineering and Remote Sensing* 63(6), 707-716.

Staufer, P. & Fischer, M. M. (1997), Spectral pattern recognition by a two-layer perceptron: effects of training set size, in I. Kanellopoulos, G. G. Wilkinson, F. Roli

& J. Austin, eds., Neurocomputation in Remote Sensing Data Analysis, Springer, Berlin - Heidelberg, 105-116.

Stenback, J. M. & Congalton, R. G. (1990), Using Thematic Mapper imagery to examine forest understory, *Photogrammetric Engineering and Remote Sensing* 56(9), 1285-1290.

Swain, P. H. & Davis, S. M., eds. (1978), Remote Sensing - the Quantitative Approach, McGraw-Hill, New York, US.

Thomas, I. L., Benning, V. M. & Ching, N. P. (1987), Classification of Remotely Sensed Images, Adam Hilger, Bristol.

Thomas, G., Hobbs, S. E. & Dufour, M. (1996), Woodland area estimation by spectral mixing: applying a goodness-of-fit solution method, *International Journal of Remote Sensing* 17(2), 291-301.

Tompkins, S., Mustard, J. F., Pieters, C. M. & Forsyth, D. W. (1997), Optimiziation of end-members for spectral mixture analysis, Remote Sensing of the Environment 59, 472- 489.

Uebele, V., Abe, S. & Lan, M. S. (1995), A neural network based fuzzy classifier, *IEEE Transactions on Systems Man and Cybernetics* 25(2), 353- 361.

Wan, E. A. (1990), Neural network classification: a Bayesian interpretation, *IEEE Transactions on Neural Networks* 1(4), 303-305.

Wang, F. (1990a), Fuzzy supervised classification of remote sensing images, *IEEE Transactions on Geoscience and Remote Sensing* 28(2), 194-201.

Wang, F. (1990b), Improving remote sensing image analysis through fuzzy information representation, *Photogrammetric Engineering and Remote Sensing* 56(8), 1163-1169.

Warner, T. A. & Shank, M. (1997), An evaluation of the potential for fuzzy classification of multispectral data using artificial neural networks, *Photogrammetric Engineering and Remote Sensing* 63(11), 1285-1294.

White, J. D., Kroh, G. C. & Pinder, J. E. (1995), Forest Mapping at Lassen Volcanic National Park, California, using Landsat TM data and a Geographical Information System, *Photogrammetric Engineering and Remote Sensing* 61(3), 299-305.

Wilkinson, G. G. (1997), Open questions in neurocomputing for earth observation, in I. Kanellopoulos, G. G. Wilkinson, F. Roli & J. Austin, eds., Neurocomputation in Remote Sensing Data Analysis, Springer, Berlin - Heidelberg, 3-13.

Wilkinson, G. & Folving, S. (1991), Fieldwork for Corine land cover updating, 9-18th june, 1991, mission report 2664, EMAP/JRC.

Wilkinson, G. G., Folving, S., Fullerton, K. & Megier, J. (1991), A study on the automatic revision of the European Community's CORINE land cover database using satellite data, in *International Archives of Photogrammetry and Remote Sensing*, Vol. XXIX of Commission IV, 543-548.

Wilkinson, G., Fullerton, K., Folving, S. & Kanellopoulos, I. (1994), Results of field studies in portugal, Technical report, EMAP/JRC.

Wilkinson, G. G., Folving, S., Kanellopoulos, I., McCormick, N., Fullerton, K. & Megier, J. (1995), Forest mapping from multi-source satellite data using neural network classifiers - an experiment in Portugal, *Remote Sensing Reviews* 12, 83-106.

Wood, T. F. & Foody, G. M. (1989), Analysis and representation of vegetation continua from Landsat TM data for lowland heaths, *International Journal of Remote Sensing* 10(1), 181-191.

Woodcock, C. E., Gopal, S. & Albert, W. (1996), Evaluation of the potential for providing secondary labels in vegetation maps, *Photogrammetric Engineering and Remote Sensing* 62(4), 393-399.

Works, G. A. (1992), Neural network basics, in P. G. J. Lisboa, ed., Neural Networks: Current Applications, Chapman & Hall, London, 35-48.

Yager, R. R. (1994), Modeling and formulating fuzzy knowledge bases using neural networks, *Neural Networks* 7, 1273-1283.

Yoo, H. & Pimmel, R. L. (1994), The effect of weight precision and range on neural network classifier performance, *Neurocomputing* 6, 541-549.

Yoshima, T. & Omatu, S. (1994), Neural network approach to land cover mapping, *IEEE Transactions on Geoscience and Remote Sensing* 32(5), 1103-1109.

Zadeh, L. A. (1965), Fuzzy sets, *Information and Control* 8, 338-353.

Zadeh, L. A. (1978), Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems* 1, 3-28.

di Zenzo, S., Bernstein, R., Degloria, S. D. & Kolsky, H. G. (1987), Gaussian maximum likelihood and contextual classification algorithms for multicrop classification', *IEEE Transactions on Geoscience and Remote Sensing* 25(6), 805-813.

Zhang, J. & Foody, G. M. (1998), A fuzzy classification of sub-urban land cover from remotely sensed imagery, *International Journal of Remote Sensing* 19(14), 2721-2738.

Zhu, A.-X. (1997), Measuring uncertainty in class assignment for natural resource maps under fuzzy logic, *Photogrammetric Engineering and Remote Sensing* 63(10), 1195-120.

Appendix A

SOFT CLASSIFICATION SOFTWARE PACKAGE

This appendix provides an overview of the functionality of the software package that was developed in the course of this thesis. For each routine, a statement of general purpose, a list of required user inputs and the type of output produced are supplied. The description of the routines is intended for the user and therefore technicalities of the implementation are not discussed. The program was written in the 'C' language on a Sun Unix system. The source code for routines which are contained in the software but which were not implemented by the author is not included in this thesis. However, changes to the original routines were listed at the top of each source file in header comments which are included in section A.4. The computer source code for the routines written by the author is listed at the end of this appendix in section A.5.

A.1 File Formats

ASCII format data files are expected. The neural network requires two training files: one containing pixel information and the other containing target information (discussed in chapter III). In the first, each row contains information for one pixel. There are as many rows as there are pixels to be trained with; there are as many columns as there are separate sources of information. For example, digital numbers from five Landsat TM channels constitute five separate sources of information. The order of the columns remains constant throughout the file. In the second training file, there are as many rows as there are pixels in the first training file. Each row represents the corresponding target information for each pixel. Targets are composed of ***n*** columns where ***n*** is the number of *pure* classes in the classification problem. Testing files follow the same convention.

Targets are created from ground data information which may consist of an integer class code, for *pure* classification, or of a ground data vector, for *soft* classification. The format of the ground data vector is: x y x y x y and so on, where x is an integer representing percentage cover and y is an integer class number. For example, a mixture of

[30% class 2 + 40% class 4 + 25% class 6 + 5% class 1] and a *pure* pixel of class 7 must be represented in the following way: '30 2 40 4 25 6 5 1' and '100 7' (without the quotes). The order of the class-percentage pairs is unimportant.

A.2 Menu Options

This section provides an overview of the routines implemented by the author which form the *soft* classification package. In addition to the menu option shown, all menus include a "q - Quit" option which stops the execution of the program and returns the user to the operating system's prompt, and all menus except the top menu include a "b - Go Back" option which takes the user from the current menu to the previous menu.

A.2.1 Upon entry

When the program is called, the user is shown the following menu.

Main Menu
1 - Prepare files for input to the neural network
2 - Neural network classification
3 - Analyse outputs from the neural network

The first option presents the user with a menu of operations to manipulate files and prepare data for input into the neural network. The second option provides a menu of neural network operations. The third option prints a menu of tools for the analysis of neural network outputs. During the execution of routines, progress is monitored by reporting the row number which is being processed. The user is asked every how many rows should reporting take place.

A.2.2 Main Menu: 1 - Prepare files for input to the neural network

The following menu is printed to the screen:

Preparation Menu

1 - Divide data into training and testing sets
2 - Modify training or testing file for input to the NNet
3 - Randomise file
4 - Divide file vertically (e.g. into DN and Class file)
5 - Create target from file of ground data
6 - Change pixel codes using a Look Up Table
7 - Join files vertically
8 - Remove pixels that have the code given in the Look Up Table
9 - Check whether ground data percentages sum to 100%
10 - Sort ground data and provide statistics

Prepare Menu: 1 - Divide data into training and testing sets

Purpose: used to divide a data file into training and testing sets. Two options are available. The data file can be divided by selecting the number of rows to place in the testing file for every row in the training file. For this option, randomisation of the data file is recommended prior to carrying out this operation. Alternatively, if each pixel has a code such as a polygon code, a look up table (LUT) can be used which contains one column of the codes for pixels which should be used as training data.

User Input:

- Name of input data file and Number of header lines to skip if any
- Option for dividing the data file ('r' by row or 'c' by code)
- Name of output training file and Name of output testing file
- If choice is 'r' then Number of ratio rows in the testing file for Number of ratio rows in the training file
- If choice is 'c' then total Number of columns in the input data file and Name of LUT file

Output:

- Training file (no header)
- Testing file (no header)

Preparation Menu: 2 - Modify training or testing file for input to the NNet

Purpose: used to centre and rescale data values so that the numbers are centred about 0. Calculates the mean and standard of each band in the training file and saves them in a log file which it then applies to the testing file. Therefore the training file must be modified before the testing file. Implemented in the original software.

User Input:

- Option of training or testing file to be modified ('t' training or 'v' verification)
- Name of input data file and Number of rows
- Option of ground data being present of not ('y' present, 'n' not present)
- Number of data bands
- Option of modifying all the bands or only some ('y' all, 'n' some)
- if choice is 'n' some, List of the bands to be modified
- if choice is 'v' testing, Name of Log file to read

Output:

- Modified training or testing file

Preparation Menu: 3 - Randomise file

Purpose: used to randomise the rows of a data file. Required for adequate learning by the neural network. The program expects no header lines for the input data file. If header lines exist they will be randomised too.

User Input:

- Name of input data file and Number of rows in the file
- Name of randomised output data file

Output:

- Randomised output file (no header)

Preparation Menu: 4 - Divide file vertically (e.g. into DN and Class file)

Purpose: used to divide a data file into two files, for example one containing only data information and the other containing the ground data information for the same pixels.

User Input:

- Name of input data file and Number of header lines to skip if any
- Number of columns to place in first file (n)
- Name of output first file and Name of output second file

Output:

- One file containing the first 'n' columns of the input data file (no header)
- A second file containing the remaining columns from the input data file (no header)

Preparation Menu: 5 - Create target from file of ground data

Calls the "Create Target Menu" which is described in section A.2.5

Preparation Menu: 6 - Change pixel codes using a Look Up Table

Purpose: used to change pixel codes to other codes or ground data. For example, it may not be possible to output anything other than an integer code for pixels when extracting them from polygons using Arc/Info. This routine can be used to match integer codes to the relevant ground data information. The LUT can contain in the first column an Arc/Info code (for example '345'). The numbers after each code can represent ground data (for example [30 1 20 4 50 8]). Alternatively, pixels from different classes can be merged

together under a new class code using an appropriate Look Up Table which assigns the same new code to different pixel codes.

User Input:

- Name of input data file and Number of header lines to skip if any
- Total Number of columns of data in the data file
- Name of LUT file and Number of header lines to skip if any
- Name of output file

Output:

- File with the same information as the input data file but with different codes or ground data information for each pixel (no header).

Preparation Menu: 7 - Join files vertically

Purpose: used to join two files column wise. All the columns from the second file are added to the first 'n' columns from the first file.

User Inputs:

- Name of first input data file and Number of header lines to skip if any
- Number of columns to include from the first file (counting from left to right)
- Name of second input data file and Number of header lines to skip if any
- Name of output file

Output:

- File containing 'n' columns from the first and all the columns from the second input files. Three header lines to the output file. Line one and two provide the name of the first and second file respectively; the third line is blank.

Preparation Menu: 8 - Remove pixels that have the code given in the Look Up Table

Purpose: used to delete pixels with a specific code. For example pixels from a specific polygon may have to be deleted because of missing information or pixels from a specific class may have to be deleted for a new classification scheme without that class.

User Inputs:

- Name of input data file and Number of header lines to skip if any
- Number of columns in this file
- Name of LUT file and Number of header lines to skip if any
- Name of output file

Output:

- File containing all the pixels from the input data file less those pixels with a code listed in the LUT file (no header).

Preparation Menu: 9 - Check whether ground data percentages sum to 100%

Purpose: used to verify that the ground data information is correct at least in terms of percentage cover by checking that they sum to 100. Assumes a file containing only ground cover information.

User Inputs:

- Name of input file and Number of header lines to delete if any

Output:

- Message "WARNING: at row 'x' the total ground coverage was 'y' or
- Message "All ground data cover sums to 100%"

Preparation Menu: 10 - Sort ground data and provide statistics

Purpose: used to sort the ground information in increasing order of percentage cover. Also produces a matrix of ground data information which, for each class, counts the number of times the class occurs in each position (dominant, secondary....). When two or more classes have the same percentage cover, they are all assigned to their highest level of component mixing. For example, if a mixture is [30% 1 + 30% 2 + 40% 3], both classes 1 and 2 will increment their counter in the secondary class. The matrix is called a 'composition' matrix and is discussed in chapter IV.

User Inputs:

- Name of ground data file and Number of header lines if any
- Number of *pure* classes
- Name of output sorted ground data file and Name of output composition matrix file

Output:

- File containing ground data information sorted, for each pixel, in increasing order of percentage cover
- File containing a matrix describing the composition of the data set

A.2.3 Main Menu: 2 - Neural network classification

The following menu is printed to the screen:

Neural Network Menu
1 - Training
2 - Testing
3 - Classification

Chapter III discusses network training, testing and classification.

Neural Network Menu: 1 - Training

Purpose: used for training a network

User Inputs:

- Learning rate, Threshold network error, Number of Iterations
- Option of initialising weights randomly 'r' or from a file 'f' (option f is for continuing training for example if want more iterations)
- if choice is 'r' (randomly) then Option of initialising with current time as a seed 'y' or with a constant 'n'
- Number of input nodes (number of bands), Number of output nodes, Number of hidden layers, Number of nodes in each hidden layer
- if choice is 'f' from a file, Name of Weights file
- Name of input Training data file and Name of input Training target file
- Number of pixels in the Training file
- Name of input Testing data file and Name of input Testing target file
- Number of pixels in the Testing file
- Root name for resulting files (all output files have this name followed by a specific extension e.g. '.wts' for weight file
- Option of producing performance matrices every 15 iterations (only if *pure* classification) ('y' yes, 'n' no)

Output:

- File of weights
- File of error values
- if choice of performance matrices is 'y' File of Training performance matrices and File of Testing performance matrices

Neural Network Menu: 2 - Testing

Purpose: used for testing the network

User Inputs:

- Name of Weights file
- Option of producing a confusion matrix of the performance (*Pure* classification only) ('y yes, 'n' no)
- Option of producing a file of neural network output values for each node for each pixel ('y' yes, 'n' no)
- Root name for output files
- Number of pixels in Testing file

Output:

- Either a confusion matrix for *pure* classification
- or/and a file of neural network output values

Neural Network Menu: 3 - Classification

Purpose: used for classifying a raw image in band interleaved by line format using the neural network. Includes the possibility of producing fraction images. The original routine was modified extensively.

User Inputs:

- Name of Image to be classified
- Number of rows and Number of columns in the image
- Name of Weight file
- Name of Output classified image
- Name of Output statistics image
- Option of creating fraction images ('y' yes, 'n' no)
- if choice is 'y' for fraction images, Option of Normalised 'n' or just scaled outputs 's' (Normalised makes sure the neural network output for each pixel sum to 1 and are scaled between 0 and 1, Scaled just scales between 0 and 1)
- Root name for outputs

Output:

- Classified image
- if choice is 'y' for fraction images then one image per class

A.2.4 Main Menu: 3 - Analyse outputs from the neural network

The following menu is printed to the screen:

Analysis Menu

1 - Create index file
2 - Calculate rank matrices
3 - Linearise Nnet outputs
4 - Scale and/or normalise and/or noise treat NNet outputs
5 - Calculate sum, max, 2nd max and difference
6 - Make pixel Ref.NNet correspondence images

The options from this menu are the implementation of the analysis tools described in chapter VI and chapter VII of the thesis.

Analysis Menu: 1 - Create index file

Purpose: used to compare the target provided to the neural network and the resulting output vector. The classes on the ground are ordered in increasing order of coverage of the pixel according to the target vector. The vector of output classes from the neural network is ordered in ascending order of coverage. A vector is produced which compares class rankings for each non-zero coverage class value. A correctly ranked class is indicated by a 1, an incorrectly ranked class is indicated by a 2. Irrelevant classes are indicated by 0.

User Input

- Name of the target file and Number of header lines to skip if any
- Name of the neural network outputs file and Number of header lines to skip if any
- Number of *pure* classes
- Value for the target vector below which ranking is not of interest (usually set to 0% coverage)
- Value for the neural network output vector below which ranking is not of interest (for example -1.5)
- Name of the output file

Output

- File -so called index file - containing rows of three vectors. First vector is that of the target with classes ordered in ascending order of coverage. The second vector is that of classes ordered in ascending order according to the neural network outputs. The third vector compares the rankings at each position.

Analysis menu: 2 - Calculate rank matrices

Purpose: Used to calculate rank misclassification matrices from the index file. Two matrices are created. One of the matrices calculates the accuracy of positioning of the classes. In other words, how many times the dominant class is correctly identified and if it is not, where is it placed. The other matrix calculates misclassifications between classes at every position, not only for the dominant class. Classes which have the same ground coverage according to the reference data are said to have the same rank. The percentage of correctly classified ranks and classes is output for each position and class. In addition, for the rank matrix, the true percentage - i.e. the number of pixels where all the positions are correctly classified is provided.

User Input:

- Name of the index file and Number of header lines to skip if any
- Number of *pure* classes
- Name of the output file

Output

- File - rank matrix file - containing two matrices. The first - rank matrix - provides information as to the positioning of classes. The second - modified misclassification - provides information as to the misclassification between classes.

Analysis Menu: 3 - Linearise NNet outputs

Purpose: used to linearise the neural network outputs. The sigmoid function is applied to the neural network outputs before their value is produced. This routine inverts its effect *a-posteriori* so that neural network values are as if no sigmoid had been applied

User Input:

- Name of the input neural network output file and Number of header lines to skip if any
- Number of *pure* classes
- Name of the output file

Output

- File containing linearised equivalents of the neural network outputs

Analysis Menu: 4 - Scale and/or normalise and.or 'noise' treat Nnet outputs

Purpose: used to scale and/or normalise and/or remove a set value from each neural

network output. Used prior to creating correspondence images. Neural network outputs lie between -1.7 and 1.7. To create the correspondence images, need outputs which sum to 1. In addition, this routine can be used to study the effect of removing small amounts from the neural network outputs, either from all of them or only from those which are less than the amount.

User Input:

- Name of the input neural network output file and Number of header lines to skip if any
- Option of scaling file 'y' or not 'n'
- If choice is yes 'y', user must provide old minimum, old maximum, new minimum and new maximum values and Name of output scaled file
- Option of normalising file 'y' or not 'n'
- If choice is yes, file is assumed scaled to positive numbers and user must input Name of output normalised file
- Option of capping outputs 'y' or not 'n'
- If choice is yes, user must input the amount to threshold by and the method. Option of renormalising 'y' or not 'n' the outputs. Name of output capped file

Output

- Depending on the choices,
- a file containing scaled outputs (usually done before normalising),
- a file containing normalised output,
- a file containing 'capped' outputs.

Analysis Menu: 5 - Calculate sum, max. 2nd max. and difference

Purpose: identifies classes with the maximum and next maximum neural network values. Prints the sum of the neural network outputs and the difference between the two maximums for each pixel.

User Input:

- Name of the input neural network outputs file and Number of header lines to skip if any
- Number of *pure* classes
- Name of output file

Output

- File containing for each pixel the sum of the outputs, the first maximum value, the second maximum value, the difference between the first and the second maximum values.

Analysis Menu: 6 - Make pixel Ref/NNet correspondence images

Purpose: produces a ground data/neural network correspondence image from a file containing the vector of ground data (e.g. target file) and the vector of neural network outputs. File produced is a raw image.

User Input:

- Name of the input image data file and Number of header lines to skip if any
- Number of *pure* classes
- Name of output image file
- Number of columns to represent 100%

Output

- Raw band interleaved by line file showing the correspondence between the ground data and the neural network outputs.

A.2.5 Preparation Menu: 5 - Create target from file of ground data

Third level menu which permits the following options

Create Target Menu

1 - Pure integer - class given as a column of integers
2 - Pure target - class info treated as if pure
3 - Scaled target - class % scaled between min and max
4 - Bin target - class % placed in bins
5 - Occurrence target - class given max. target value if it occurs
6 - Rescale vector - vector components scaled one by one

The options in this menu implement the types of targets described in chapter VII of the thesis.

Create Target menu: 1 - Pure integer - class given as a column of integers

Purpose: Routine which assumes as input a file containing a column of integers indicating the *pure* class to which a pixel belongs. Target created is for *pure* classification and contains no mixture information. Class numbers must be sequential. and start at 1.

User Input:

- Name of the input class number file and Number of header lines to skip

- Number of *pure* classes
- Value to give to the class to which the pixel belongs (usually +1)
- Value to give to the classes to which the pixel does not belong (usually -1)
- Name of output target file

Output

- File containing targets

Create Target Menu: 2 - Pure target - class info data treated as if pure

Purpose: Routine which assumes a file of ground data as input (in the format % cl % cl etc.) but which creates targets which do not take the information into consideration. Only the dominant class is considered.

User Input:

- Name of the input class number file and Number of header lines to skip if any
- Number of *pure* classes
- Value to give to the class to which the pixel belongs (usually +1)
- Value to give to the classes to which the pixel does not belong (usually -1)
- Name of output target file

Output

- File containing targets

Create Target: 3 - Scaled Target - class % scaled between min. and max.

Purpose: Routine which assumes a file of ground data as input (in the format % cl % cl etc.) which creates targets of scaled values (usually ground data is scaled between 0 and 1 say, so that 0% coverage is assigned 0 and 100% coverage is assigned 1).

User Input:

- Name of the input class number file and Number of header lines to skip
- Number of *pure* classes
- Value to give to the class to which the pixel belongs (usually +1)
- Value to give to the classes to which the pixel does not belong (usually 0)
- Name of output target file

Output

- File containing targets

Create Target Menu: 4 - Bin target - class % placed in bins

Purpose: Routine used to produce targets where each class is placed in a 'bin' on the basis of its coverage of the pixel. User must input how many bins they are, what value they take on and the percentage range which should be assigned to them.

User Input:

- Name of the input class number file and Number of header lines to skip
- Number of *pure* classes
- Number of bins
- For each bin, its value
- For each bin, the range which should be assigned to it

Output

- File containing targets

Create Target Menu: 5 - Occurrence target - class given max. target value if it occurs

Purpose: Routine which creates a target by assigning the maximum value (+1) to all classes which are present in the mixture, regardless of actual percentage cover, and minimum value (-1) to all classes which are not present in the mixture. In effect, this type of target signals the presence or not of a class

User Input:

- Name of the input class number file and Number of header lines to skip
- Number of *pure* classes
- Value to give to the class to which the pixel belongs (usually +1)
- Value to give to the classes to which the pixel does not belong (usually 0)
- Name of output target file

Output

- File containing targets.

Create Target menu: 6 - Rescale Vector - vector components scaled one by one

Purpose: Routine used to rescale a vector's values 1 by 1. i.e.with different min.'s and max.'s for each. Mainly used in the biophysical experiments to scale back from the neural network outputs to height and basal area value - for example.

User Input:

- Name of the input class number file and Number of header lines to skip

- Number of *pure* classes
- New minimum
- New maximum
- Old minimum
- Old maximum

Output

- File containing targets.

A.3 Miscellaneous

The code is organised into

- one file containing routines determining menu choices (*NNsoft_Main.c*),
- one file for preparation routines (*NNsoft_Prepare.c*),
- one file for analysis routines (*NNsoft_Analyse.c*),
- one file for routines to create targets (*NNsoft_CreateTar.c*) and
- one file for routines which complement the routines originally implemented (*NNsoft_MyAdditions.c*).
- A number of files for routines containing the neural network software

Figure A-1 shows the location of routines within each source code file and the routines from which calls to other routines are made. The user need not know this information.

Alternatives to the routines exist. For example, the randomise routine can be replaced by sorting using for example a UNIX command; division of data files can be executed in awk, and so on.

Logical Flow for Soft Classification Software Functions

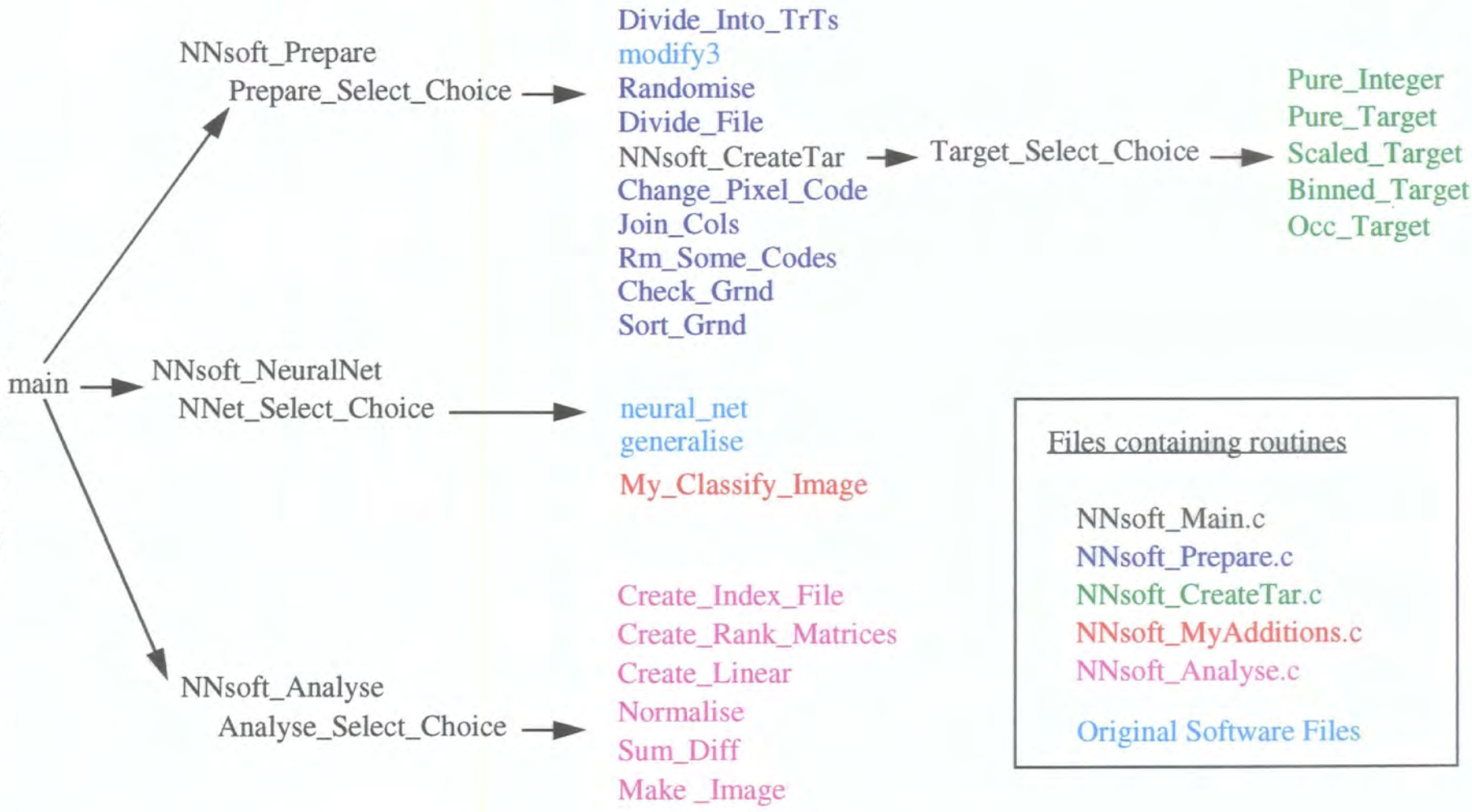


Figure A-1. Location of routines within software source files

Header

```
/*****************************************************
* FILE: NNsoft_modify3.c                             *
*                                                    *
* DESCRIPTION:                                       *
* Computes the means and standard deviations for     *
* each channel for centering the pixel values of a   *
* training data file.                                *
* The means and standard deviations are saved in a   *
* log file which is then used to centre the pixel    *
* values of a verification set and/or an image file. *
* The new files are created in a format suitable     *
* for input in a neural network. (17/5/91)           *
*                                                    *
* AUTHOR: I. Kanellopoulos (JRC)                     *
*                                                    *
* LIST of routines:  modify3                         *
*                    user_session                    *
*                    readfile                        *
*                    writefile                       *
*                    scaledown                       *
*                    modifyfile                      *
*                                                    *
* NEW routines: NONE                                 *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_utils.c:          dalloc                    *
*                          double_matrix             *
*                          char_matrix               *
*                          free_d_matrix             *
*                          free_c_matrix             *
* NNsoft_My_Additions.c: My_File_Open                *
*                          My_Get_File               *
*                                                    *
* LATEST modification: 16-05-1998                    *
*                                                    *
* MODIFICATIONS: A.C. Bernard (JRC/Durham)           *
* - Declarations changed to ANSII C standard         *
* - #include and some function declarations removed  *
*   now in NNsoft_neural.h                           *
* - file_open function changed to My_File_Open and   *
*   My_Get_File                                      *
* - Constant values changed to constant definitions  *
*   in NNsoft_neural.h                               *
* - File *fpname declarations moved to within        *
*   routines rather than global                      *
* - When questions asked, relooping if invalid       *
*   answer implemented                               *
* - Use of tolower                                   *
* - Some slight restructuring of some if loops e.g.  *
*   when some function called in both arms of the    *
*   if condition, removed to outside                 *
* - All use of class as an integer array changed to  *
*   grnd_data - a character array                    *
* - Naming convention changed. No longer ask for     *
*   filename without extension, ask for full name to *
*   which add '.mod' for modified file and '.log'    *
*   for logname.                                     *
* - Header comments added (11-12-1997)               *
*                                                    *
* - missing return statements added (16-05-1889)     *
*                                                    *
*****************************************************/
/*****************************************************
* FILE: NNsoft_gradient.c                            *
*                                                    *
* DESCRIPTION:                                       *
* Part of the neural network program where training  *
* is carried out.                                    *
*                                                    *
* AUTHOR: I. Kanellopoulos (JRC)                     *
*                                                    *
* LIST of routines: gradient_descent                 *
*                   system_status                    *
*                   compute_current_error            *
*                                                    *
* NEW routines: NONE                                 *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_lipclassify.c : write_error                 *
*                        write_header                *
*                        test_quality                *
* NNsoft_netarch.c     : write_weights               *
* NNsoft_neuralnet.c   : forward_pass                *
*                        seteta                      *
*                                                    *
* LATEST Modification: 16-05-1998                    *
*                                                    *
* MODIFICATIONS - A. C. Bernard  (JRC-Durham)        *
* - Removed declaration of routines and variables,   *
*   now in NNsoft_neural.h                           *
* - Error filename now assigned in NNsoft_netarch.c  *
*   no longer here                                   *
* - NUMBER_OF_EPOCHS is no longer a variable but is  *
*   a defined constant in NNsoft_neural.h            *
* - No longer have the 'optimisation' feature of     *
*   if < -1.0 = -1.0 and if > 0.9 = 1.0 as unsure of *
*   effect when doing FUZZY classification           *
* - Added header to error file                       *
* - Error file closing done in write_error routine   *
*   removed from here                                *
* - Used definition ans[SMALL] for y/n answer        *
*   instead of answer[10]                            *
* - No longer print gac and lac to error file        *
*   because make no sense with FUZZY classification  *
* - Test_quality changed from returning a double     *
*   to returning void, no usage changed here         *
* - Header comments added   (10-12-1997)             *
*                                                    *
* - missing return statements added (16-05-1998)     *
*                                                    *
*****************************************************/
/*****************************************************
* FILE: NNsoft_lipclassify.c                         *
*                                                    *
* DESCRIPTION:                                       *
* Performs classification of images which are  in    *
* ERDAS/LIP. The image data is read with a specified *
* number of lines each time and each pixel is        *
* centred using the means and variances which are    *
* saved in a log file. Also tables for the confusion *
* matrices are written in files for both the         *
* training set and verification set.                 *
*                                                    *
* Now runs with raw images - i.e. images which have  *
* no header etc, just rows and columns of pixels     *
* The classification routine is located in           *
* NNsoft_MyAdditions.c (A.C.Bernard)                 *
*                                                    *
* AUTHOR: I. Kanellopoulos (JRC)                     *
*                                                    *
* LIST of routines: generalisation                   *
*                   read_log                         *
*                   write_result                     *
*                   write_header                     *
*                   write_error                      *
*                   test_quality                     *
*                   classify_image                   *
*                                                    *
* NEW routines: NONE                                 *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_netarch.c:       archread                   *
*                         read_weights               *
* NNsoft_neural_net.c:    initialise                 *
*                         forward_pass               *
* NNsoft_MyAdditions.c :  My_File_Open               *
*                         My_Get_File                *
*                         My_Write_Result            *
* NNsoft_utils.c:         dalloc                     *
*                         int_matrix                 *
*                         double_matrix              *
*                         free_d_matrix              *
*                         free_i_matrix              *
*                                                    *
* LATEST modification: 16-05-1998                    *
*                                                    *
* MODIFICATIONS - A.C.Bernard (JRC-Durham)           *
* - Routine declarations/definitions changed to ANSI *
*   C standard                                       *
* - All declarations of routines etc moved to        *
*   NNsoft_Neural.h                                  *
* - var[int] usage changed to var[CONSTANT] where    *
*   CONSTANT is defined in NNsoft_Neural.h           *
* - Some filenames and pointer names changed to more *
*   meaningful names                                 *
* - Filenames and pointers changed to local variable *
*   instead of global variables wherever possible    *
* - Classification option menu moved to NNsoft_Main. *
```

Header

```
* - Looping added if invalid type of answer         *
* - Full filenames must be entered, automatic       *
*   assumption of .dat extension no longer valid nor*
*   verified nor is .log extension assumed          *
* - classid usage removed since target now read from *
*   separate file                                   *
* - Average omit and commit variables no longer      *
*   calculated                                      *
* - classify_image routine commented out, routine to *
*   be used is My_Classify_Image in                 *
*   NNsoft_MyAdditions.c                            *
* - Option to produce confusion matrix as opposed to *
*   de facto production added as well as option to  *
*   produce file of neural network values           *
* - header added to resulting files                  *
* - some file closing and freeing of memory added    *
* - write_result routine return value void instead   *
*   of double                                       *
* - write_header contents changed, idea the same     *
* - Added Cycle number (NUMOFEPOCHS)                 *
* - test_quality changed from this declaration:      *
*   dble tst_qlty(npatterns, filename, result_file)  *
*   to vd tst_qlty(npatterns, indata_name,           *
*   intrgt_name, outres_name)                        *
* - Header comments added (05-01-1998)               *
*                                                    *
* - missing return statements added (16-05-1998)     *
*                                                    *
*****************************************************/
/*****************************************************
* FILE: NNsoft_netarch.c                             *
*                                                    *
* DESCRIPTION:                                       *
* Part of the neural network program concerned with  *
* obtaining achitecture and parameter information    *
*                                                    *
* AUTHOR: I. Kanellopoulos                           *
*                                                    *
* LIST of routines:   read_parameters                *
*                     read_net_arch                  *
*                     read_train_file                *
*                     write_weights                  *
*                     archread                       *
*                     read_weights                   *
*                                                    *
* NEW routines: NONE                                 *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_lipclassify.c:  write_header                *
* NNsoft_utils.c:        double_matrix               *
*                        get_double                  *
*                        dalloc                      *
*                        illoc                       *
* NNsoft_My_Additions.c: My_File_Open                *
*                        My_Get_File                 *
*                                                    *
* LATEST modification: 16-05-1998                    *
*                                                    *
* MODIFICATIONS - A.C. Bernard (JRC/Durham)          *
* - Removed option of default values for parameters  *
* - Write out architecture chosen                    *
* - Changed some filenames to more meaningful names  *
* - Made file pointers and names local rather than   *
*   global                                           *
* - Looping if invalid type of answer                *
* - Added header to resulting files                  *
* - Option to produce partial matrices rather than   *
*   de facto production                              *
* - Commented out any use of a-priori                *
* - Momentum term not used                           *
* - int c declarations and use removed since read    *
*   target from different file                       *
* - No more directory path option, nor assumption of *
*   .dat extensions                                  *
* - Target read from a separate file to the data     *
* - Some extra closing of files                      *
* - Header comments added (05-01-1998)               *
*                                                    *
* - removed threshold of unclassified pixels since   *
*   used nowhere (12-03-1998)                        *
*                                                    *
* - added missing return statements (16-05-1998)     *
*                                                    *
*****************************************************/
/*****************************************************
* FILE: NNsoft_neural_net.c                          *
*                                                    *
* DESCRIPTION:                                       *
* Part of the neural network program where training  *
* is started out, info obtained, etc.                *
*                                                    *
* AUTHOR: I. Kanellopoulos (JRC)                     *
*                                                    *
* LIST of routines: neural_net                       *
*                   initialise                       *
*                   init_weights                     *
*                   forward_pass                     *
*                   scale_01                         *
*                   a_priori                         *
*                   scale_back                       *
*                   set_eta                          *
*                   onintr                           *
*                                                    *
* NEW routines: NONE                                 *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_gradient.c:     compute_current_error       *
* NNsoft_netarch.c:      netarch                     *
*                        archread                    *
*                        read_train_file             *
*                        read_weights                *
*                        read_parameters             *
*                        read_net_arch               *
* NNsoft_utils.c:        free_d_matrix               *
*                        dalloc                      *
* NNsoft_MyAdditions.c: My_Init_Weights              *
*                                                    *
* LATEST modification: 16-05-1998                    *
*                                                    *
* MODIFICATIONS - A. C. Bernard (JRC/Durham)         *
* - Menu options moved to NNsoft_Main.c              *
* - Added option of seeding weights with time rather *
*   than a constant                                  *
* - For all questions, added looping if invalid      *
*   type of answer                                   *
* - Header comments added (06-01-1998)               *
*                                                    *
* - missing return statements added (16-05-1998)     *
*                                                    *
*****************************************************/
/*****************************************************
* FILE: NNsoft_utils.c                               *
*                                                    *
* DESCRIPTION:                                       *
* Contains utility routines for allocating memory    *
* and freeing memory and a few others                *
*                                                    *
* AUTHOR: I. Kanellopoulos  (JRC)                    *
*                                                    *
* LIST of routines: ulloc                            *
*                   illoc                            *
*                   falloc                           *
*                   dalloc                           *
*                   Byte_matrix                      *
*                   int_matrix                       *
*                   float_matrix                     *
*                   double_matrix                    *
*                   free_Byte_matrix                 *
*                   free_i_matrix                    *
*                   free_f_matrix                    *
*                   free_d_matrix                    *
*                   get_int                          *
*                   get_double                       *
*                   extension                        *
*                   file_open                        *
*                                                    *
* NEW routines:     get_file (moved from original    *
*                             location netarch.c)    *
*                                                    *
* DEPENDENCIES: NONE                                 *
*                                                    *
* LATEST Modification: 16-05-1998                    *
*                                                    *
* MODIFICATIONS - A. C. Bernard (JRC-Durham)         *
* - Routine definitions changed to ANSII C standard  *
*   format                                           *
* - All #define, variable and routine declarations   *
*   etc. moved to NNsoft_neural.h file               *
```

```
/*****************************************************
* FILE: NNsoft_Main.c                                *
*                                                    *
* DESCRIPTION:                                       *
*    Top level routines that control                 *
*    the choices of the three basic                  *
*    modules:                                        *
*    A- file manipulations                           *
*    B- neural network                               *
*    C- analysis routines                            *
*                                                    *
* AUTHOR: A. C. Bernard                              *
*                                                    *
* LIST of routines:  Main                            *
*                    Main_Select_Choice              *
*                    NNsoft_Prepare                  *
*                    Prepare_Select_Choice           *
*                    NNsoft_Analyse                  *
*                    Analyse_Select_Choice           *
*                    NNsoft_NeuralNet                *
*                    NNet_Select_Choice              *
*                    NNsoft_CreateTar                *
*                    Target_Select_Choice            *
*                                                    *
* DEPENDENCIES: NONE                                 *
*                                                    *
* MODIFICATIONS:                                     *
*****************************************************/

@includo "NNsoft_neural.h"


/************ MAIN ************/
main(void)
{
  int choice;


/* Print the header of the program and activate the function
   which prints the Main Menu */
printf("\n\n*************************************************************\n");
printf("*  Neural Network Soft Classification Program  *\n");
printf("*                                              *\n");
printf("*                                              *\n");
printf("*          (c) IRC-SAI-EMAP, 1991-1998 *\n");
printf("*************************************************************\n\n");


/* Repeat the menu until EXIT is chosen. Depending on
   the choice, activate the relative function. If the
   choices are not 1,2,3 or q (=-1) the choice is
   invalid */
for (;;)
{
  choice = Main_Select_Choice();

  switch (choice)
  {
    case 1 : NNsoft_Prepare();
             break;
    case 2 : NNsoft_NeuralNet();
             break;
    case 3 : NNsoft_Analyse();
             break;
    case -1 : exit(0);

    default : printf("\nInvalid Choice ....\n");
  }

} /* End of for (;;) loop */
} /* END of main routine */


/************  SUBROUTINE: Main_Select_Choice ************/
Main_Select_Choice()
{
  char chosen[SMALL];

/* Print the Main Menu to screen */
printf("\n\n  MAIN MENU\n\n");
printf(" 1 : Prepare files for input to the neural network\n");
printf(" 2 : Neural network classification\n");
printf(" 3 : Analyse outputs from the neural network\n");
printf(" q : Quit\n\n");
printf("ENTER selection (1-3 or q) > ");


/* Read in the choice */
scanf("%s%*c",chosen);


/* If quit was chosen then set the return value to -1
   otherwise return the integer equivalent of the choice
   NOTE that if the user inputs -1 will assume that have
   input 'q' or 'Q' */
if (tolower(*chosen) == 'q')
  return (-1);
else
{
  printf("\nREPORT progress every how many rows? > ");
  scanf("%d", &EVERY_ROW);
  return atoi(chosen);
}


} /* END of Main_Select_Choice subroutine */


/************  SUBROUTINE:  NNsoft_Prepare ************/
void NNsoft_Prepare()
{
  int choice;


/* Repeat the menu until EXIT is chosen. Depending on
   the choice, activate the relative function. If the
   choices are not 1-11 or q (=-1) or b (=-2) the choice is
   invalid. Choice b brings the user up one menu */
for (;;)
{
  choice = Prepare_Select_Choice();
  switch (choice)
  {

    case -2 : return;
    case 1 : Divide_Into_TrTs();
             break;
    case 2 : modify3();
             break;
    case 3 : Randomise();
             break;
    case 4 : Divide_File();
             break;
    case 5 : NNsoft_CreateTar();
             break;
    case 6 : Change_Pixel_Codes();
             break;
    case 7 : Join_Cols();
             break;
    case 8 : Rm_Some_Codes();
             break;
    case 9 : Check_Grnd();
             break;
    case 10 : Sort_Grnd();
              break;
    case -1 : exit(0);

    default : printf("\nInvalid Choice ....\n");
  }

} /* End of for (;;) loop */
} /* END of NNsoft_Prepare subroutine*/


/************  SUBROUTINE: Prepare_Select_Choice ************/
Prepare_Select_Choice()
{
  char chosen[SMALL];


/* Print the Preparation Menu to the screen */
printf("\n\n  PREPARATION MENU\n\n");
printf(" b : Go Back\n");
printf(" 1 : Divide data into training and testing sets\n");
```



```
printf(" 2 : Modify training or testing file for input to the NNet\n");
printf(" 3 : Randomise file\n");
printf(" 4 : Divide file vertically (e.g. into DN file and Class file) \n");
printf(" 5 : Create target from file of ground data\n");
printf(" 6 : Change pixel codes using a Look Up Table\n");
printf(" 7 : Join files vertically \n");
printf(" 8 : Remove pixels that have the code given in the Look Up Table \n");
printf(" 9 : Check whether ground data percentages sum to 100%%\n");
printf(" 10 : Sort ground data and produce statistics\n");
printf(" q : Quit\n\n");
printf("ENTER selection (1-10, b or q) > ");


/* Read in choice */
scanf("%s%*c",chosen);


/* If quit was chosen then set the return value to -1 else if
   back was chosen then set the return value to -2,
   otherwise return the integer equivalent of the choice
   NOTE that if the user inputs -1 will assume that have
   input 'q' or 'Q' and similarly for -2 */
if (tolower(*chosen) == 'q')
  return (-1);
else if (tolower(*chosen) == 'b')
  return (-2);
else
  return atoi(chosen);


} /* End of Prepare_Select_Choice subroutine */


/************  SUBROUTINE: NNsoft_NeuralNet ************/
void NNsoft_NeuralNet(void)
{
  int choice =0;


/* Repeat the menu until EXIT is chosen. Depending on
   the choice, activate the relative function. If the
   choices are not 0-11 or q (=-1) or b (=-2) the choice is
   invalid. Choice 0 brings the user up one menu */
for (;;)
{
  choice = NNet_Select_Choice();

  switch (choice)
  {

    case -2 : return;
    case 1 : neural_net();
             break;
    case 2 : generalisation();
             break;
    case 3 : My_Classify_Image();
             break;
    case -1 : exit(0);
    default : printf(" Invalid Choice ....\n");
  }

} /* End of for (;;) loop */
} /* End of NNsoft_NeuralNet subroutine */


/************  SUBROUTINE: NNet_Select_Choice ************/
NNet_Select_Choice()
{
  char chosen[SMALL];


/* Print Neural Network Menu to the screen */
printf("\n\n  NEURAL NETWORK MENU\n\n");
printf(" b : Go Back\n");
printf(" 1 : Training\n");
printf(" 2 : Testing\n");
printf(" 3 : Classification\n");
printf(" q : Quit\n\n");
printf("ENTER selection (1-3, b or q) > ");


/* Read in choice */
scanf("%s%*c",chosen);


/* If quit was chosen then set the return value to -1 else if
   back was chosen then set the return value to -2,
   otherwise return the integer equivalent of the choice
   NOTE that if the user inputs -1 will assume that have
   input 'q' or 'Q' and similarly for -2 */
if (tolower(*chosen) == 'q')
  return (-1);
else if (tolower(*chosen) == 'b')
  return (-2);
else
  return atoi(chosen);


} /* End of NNet_Select_Choice subroutine */


/************  SUBROUTINE: NNsoft_Analyse ************/
void NNsoft_Analyse(void)
{
  int choice=0;


/* Repeat the menu until EXIT is chosen. Depending on
   the choice, activate the relative function. If the
   choices are not 0-11 or q (=-1) or b (=-2) the choice is
   invalid. Choice 0 brings the user up one menu */
for (;;)
{
  choice = Analyse_Select_Choice();
  switch (choice)
  {

    case -2 : return;
    case 1 : Create_Index_File();
             break;
    case 2 : Create_Rank_Matrices();
             break;
    case 3 : Create_Linear();
             break;
    case 4 : Normalise();
             break;
    case 5 : Sum_Diff();
             break;
    case 6 : Make_Image();
             break;
    case -1 : exit(0);

    default : printf("\nInvalid Choice ....\n");
  }

} /* End of for (;;) loop */
} /* END of NNsoft_Analyse subroutine*/


/************  SUBROUTINE: Analyse_Select_Choice ************/
Analyse_Select_Choice()
{
   char chosen[SMALL];


/* Print Analysis menu to the screen */
printf("\n\n  ANALYSIS MENU\n\n");
printf(" b : Go Back\n");
printf(" 1 : Create index file\n");
printf(" 2 : Calculate rank matrices\n");
printf(" 3 : Linearise NNet outputs \n");
printf(" 4 : Scale and/or normalise and/or 'noise' treat NNet outputs\n");
printf(" 5 : Calculate sum, max, 2nd max, and difference\n");
printf(" 6 : Make pixel Ref/NNet correspondance images\n");
printf(" q : Quit\n");
printf("ENTER selection (1-6, b or q) > ");


/* Read in choice */
scanf("%s%*c",chosen);


/* If quit was chosen then set the return value to -1 else if
   back was chosen then set the return value to -2,
```

```
   otherwise return the integer equivalent of the choice
   NOTE that if the user inputs -1 will assume that have
   input 'q' or 'Q' and similarly for -2 */
if (tolower(*chosen) == 'q')
  return (-1);
else if (tolower(*chosen) == 'b')
  return (-2);
else
  return atoi(chosen);

} /* END of Analyse_Select_Choice subroutine*/


/*############ SUBROUTINE: NNsoft_CreateTar ############*/
void NNsoft_CreateTar(void)
{
  int choice =0;

/* Repeat the menu until EXIT is chosen. Depending on
   the choice, activate the relative function. If the
   choices are not 0-11 or q (=-1) or b (=-2) the choice is
   invalid. Choice 0 brings the user up one menu */
for (;;)
{
  choice = Target_Select_Choice();

  switch (choice)
  {
    case -2 : return;
    case 1 : Pure_Integer();
             break;
    case 2 : Pure_Target();
             break;
    case 3 : Scaled_Target();
             break;
    case 4 : Binned_Target();
             break;
    case 5 : Occ_Target();
             break;
    case 6 : Rescale_Vector();
             break;
    case -1 : exit(0);

    default : printf("\nInvalid Choice ....\n");
  }

} /* End of for (;;) loop */
} /* END of NNsoft_CreateTar subroutine */


/*############ SUBROUTINE: Target_Select_Choice ############*/
Target_Select_Choice()
{
  char chosen[SMALL];


/* Print Target Creation Menu to the screen */
printf("\n\n   CREATE TARGET submenu\n\n");
printf(" b : Go Back\n");
printf(" 1 : Pure integer    - class given as a columo of integers\n");
printf(" 2 : Pure target     - class info data treated as if pure\n");
printf(" 3 : Scaled target   - class %% scaled between min and max\n");
printf(" 4 : Binned target   - class %% placed in 'bins'\n");
printf(" 5 : Occurence target - class given max. target value if it occurs \n");
printf(" 6 : Rescale vector  - vector components scaled one by one\n");
printf(" q : Quit\n\n");
printf("ENTER selection (1-6, b or q) > ");


/* Read in Choice */
scanf("%s%*c",chosen);


/* If quit was chosen then set the return value to -1 else if
   back was chosen then set the return value to -2,
   otherwise return the integer equivalent of the choice
   NOTE that if the user inputs -1 will assume that have
   input 'q' or 'Q'  and similarly for -2*/
if (tolower(*chosen) == 'q')
  return (-1);
else if (tolower(*chosen) == 'b')
  return (-2);
else
  return atoi(chosen);


} /* END of Target_Select_Choice subroutine */
```

```
/*****************************************************
* FILE: NNsoft_Prepare.c                             *
*                                                    *
* DESCRIPTION:                                       *
* Module of program NNsoft programs used to prepare  *
* files for use with the neural network.             *
*                                                    *
* AUTHOR: A. C. Bernard                              *
*                                                    *
* LIST of routines:   Divide_Into_TrTs               *
*                     Randomise                      *
*                     Divide_File                    *
*                     Change_Pixel_Codes             *
*                     Join_Cols                      *
*                     Rm_Some_Codes                  *
*                     Check_Grnd                     *
*                     Sort_Grnd                      *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_MyAdditions.c : My_File_Open, My_Get_File   *
*                        My_Write_Result             *
* NNsoft_utils.c       : dalloc, double_matrix       *
*                        free, free_d_matrix         *
*                        free_i_matrix               *
*                                                    *
* MODIFICATIONS:                                     *
*                                                    *
*****************************************************/


#include "NNsoft_neural.h"


/*############ SUBROUTINE: Divide_Into_TrTs ############*/
void Divide_Into_TrTs(void)
{
/* Routine used to divide a file into training and testing sets.
   Division can be done by row ratio or by looking up the codes
   of the pixels which should be placed in the training set */

  FILE   *fpout_tr, *fpout_ts, *fpin_file, *fpin_lu;
  char   outtr_name[MAX_FILENAME], outts_name[MAX_FILENAME];
  char   infile_name[MAX_FILENAME], inlu_name[MAX_FILENAME];
  char   code[MAX_STRING_SIZE], lu_code[MAX_STRING_SIZE];
  char   ans, inputs[MAX_STRING_SIZE];
  double *input_value;
  int    nrtr, nrts, num_cols, row, col, row_num=0;


/* Open files and obtain information
   del_header is used to ignore any header lines */
fpin_file=My_Get_File("\nINPUT: Data file > ", infile_name, "r");

del_header(fpin_file);


/* Find out the choice of the user: row ratio or codes */
for (;;)
{
  printf("\nDivide into Training and Testing file by");
  printf(" row ratio (R) or by code (C)? > ");
  scanf("%c%*c", &ans);
  if (tolower(ans) == 'r' || tolower(ans) == 'c')
    {break;}
  printf("Invalid choice ....\n");
}

fpout_tr=My_Get_File("\nOUTPUT: TRaining file > ", outtr_name, "w");

fpout_ts=My_Get_File("\nOUTPUT: TeSting file > ", outts_name, "w");




/* If the choice is by row ratio, then do the following.
   The assumption is that the file has been randomised previously */
if (tolower(ans) == 'r')
{
/* Find out information for division */
  printf("\nNumber of ratio rows in the TRaining file? > ");
  scanf("%d%*c",&nrtr);
  printf("\nNumber of ratio rows in the TeSting file? > ");
  scanf("%d%*c",&nrts);


printf("\nDividing...\n");

/* while the eof character of the input file
   has not been reached...then do */
for(;;)
{

/* If the EOF character has been reached then come out */
  if (feof(fpin_file))
    {break;}

/* For number of training rows, print the rows to
   the training file. Check the end of file at every row */
    for (row = 0; row < nrtr; row ++)
    {
      if (!feof(fpin_file))
      {
         fgets(inputs,MAX_STRING_SIZE, fpin_file);
         fscanf(fpin_file,"\n");
         fputs(inputs, fpout_tr);

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
        row_num += 1;
        if (row_num%EVERY_ROW == 0)
          {printf("Row = %d\n", row_num);}

      } /* End of if (!feof) condition */
    } /* End of for (row) loop */

/* For number of testing rows, print the rows to
   the testing file */
    for (row = 0; row < nrts; row ++)
    {
      if (!feof(fpin_file))
      {
        fgets(inputs,MAX_STRING_SIZE, fpin_file);
        fscanf(fpin_file, "\n");
        fputs(inputs, fpout_ts);

 /* Keep track of progress by printing the row
    number when it is a multiple of EVERY_ROW */
        row_num += 1;
        if (row_num%EVERY_ROW == 0)
          {printf("Row = %d\n", row_num);}

      } /* End of if (!feof) condition */
    } /* End of for (row) loop */

  } /* End of while (!feof) loop */

}/* End of if 'r' part of condition. Start of if 'c' part */

/* else if (tolower(*ans) == 'c') and choice is therefore by Code */

else
{
/* Obtain relevant information */
  printf ("\nNumber of columns in the input file ? > ");
  scanf("%d%*c",&num_cols);

  fpin_lu= My_Get_File("\nINPUT: Name of L,U file (codes for TRaining file) > ", inlu_name, "r");


/* Allocate memory */
  input_value= dalloc(num_cols - 1);

  printf("\nDividing...\n");

  for(;;)
  {

/* If the EOF character has been reached then come out */
  if (feof(fpin_file))
    {break;}

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
    row_num += 1;
    if (row_num%EVERY_ROW == 0)
      {printf("Row = %d\n", row_num);}
```



```
/* Read each input data source column and then the code,
   print the input data source to the new file */
    for(col = 0; col < num_cols - 1; col ++)
      {fscanf(fpin_file, "%lf%*c", &input_value[col]);}
    fscanf(fpin_file,"%s%*c", code);

/* If the EOF character has been reached then come out */
  if (feof(fpin_file))
    {break;}

/* while old class in the input file and in the LU file
  are not equal and EOF in LU hasn't been reached, read
  the new information (class or ground data) */

    *lu_code = 'x' ;
    while (strcmp(lu_code, code) != 0 && !feof(fpin_lu))
      {fscanf(fpin_lu,"%s%*c",lu_code);}


/* if class was found in the LU file print row to training file
   otherwise (eof reached) print the row in the testing file */
    if(strcmp(code,lu_code) == 0)
    {
      for(col = 0; col < num_cols - 1; col ++)
        {fprintf(fpout_tr,"%8.6lf ",input_value[col]);}
      fprintf(fpout_tr," %s\n", code);
    }
    else
    {
      for(col = 0; col < num_cols - 1; col ++)
        {fprintf(fpout_ts,"%8.6lf ",input_value[col]);}
      fprintf(fpout_ts," %s\n",code);
    } /* End of if (strcmp) condition */

   rewind(fpin_lu);

  }/* End of while !feof loop */

} /* End of if 'r' or 'c' condition */


/* Free arrays and close files */
if (tolower(ans) == 'c')
{
  free(input_value);
  fclose(fpin_lu);
}
fclose(fpin_file);
fclose(fpout_tr);
fclose(fpout_ts);

printf(" \nPREPARE : 1 - Divide data into training and testing sets...COMPLETE\n\n");

return;

} /* END of Divide_Into_TrTs subroutine */



/*############ SUBROUTINE: Randomise ############*/
void Randomise(void)
{
/* Routine used to randomise a file's rows. Uses random numbers
   which it then sorts. Alternative is to use
   system(sort -x.y filenamein > filename out) but
   this is dependant on the operating system therefore less
   portable */

  FILE   *fpin_file, *fpout_new;
  char   infile_name[MAX_FILENAME], outnew_name[MAX_FILENAME];
  char   **inputs;
  int    *inputs_ran, *inputs_index;
  int    rand_num, row_num=0, row, num_rows, i=0;
  time_t t;


/* Open files and obtain information  */
fpin_file = My_Get_File("\nINPUT: Data file > ", infile_name, "r");

printf("\nNumber of rows ? > ");
scanf("%d%*c", &num_rows);

fpout_new = My_Get_File("\nOUTPUT: Randomised data file > ", outnew_name,"w");


/* Allocate memory */
inputs = char_matrix(num_rows+1,MAX_STRING_SIZE);
inputs_ran=illoc(num_rows+1);
inputs_index = illoc(num_rows+1);


/* Seed random numbers with time */
printf("\nRandomising...\n");

srand((unsigned int) time(&t));

inputs[i] = "0";
while (!feof(fpin_file))
{
    i += 1;
    fgets(inputs[i],MAX_STRING_SIZE, fpin_file);

}

/* Add a random number to each line */
for(row=0;row< num_rows;row++)
{
  rand_num = rand();
  inputs_ran[row]=rand_num;
}


/* sort the lines in ascending order of random number */
indexx(num_rows, inputs_ran, inputs_index);

/* print each line in this new order, without the random number */
for( row = 1; row <= num_rows; row++)

{
/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

  fputs(inputs[inputs_index[row]], fpout_new);
}


/* Free arrays and close files */
free(inputs_ran);
free(inputs_index);
free_c_matrix(inputs, num_rows);

fclose(fpin_file);
fclose(fpout_new);

printf(" \nPREPARE : 3 - Randomise file...COMPLETE\n\n");

return;

}/* END of Randomise subroutine */



/*############ SUBROUTINE: Divide_File ############*/
void Divide_File(void)
{
/* Routine used to divide file vertically, i.e. put x
columns in one file and y columns in another.  Used
mainly for dividing into data and target files */

  FILE   *fpin_file, *fpout_dn, *fpout_cl;
  char   infile_name[MAX_FILENAME], outdn_name[MAX_FILENAME];
  char   outcl_name[MAX_FILENAME], *grnd_data;
  double input_value;
  int    num_cols, col, i, row_num=0;


/* Obtain filename and information
       del header used to ingore any header line */
fpin_file = My_Get_File("\nINPUT: Data file (e.g. file.mod) > ", infile_name, "r");

del_header(fpin_file);

printf("\nNumber of cols in first file > ");
scanf("%d%*c",&num_cols);
```

```
fpout_dn= My_Get_File("\nOUTPUT: First file (e.g. DN) > ", outdn_name, "w");

fpout_cl= My_Get_File("\nOUTPUT: Second file (e.g. class) > ", outcl_name,"w");


/* Allocate memory and initialise arrays */
grnd_data = clloc(MAX_STRING_SIZE);
for (i=0; i<MAX_STRING_SIZE; i++)
  grnd_data[i] = '0';

printf("\nDividing...\n");

for (;;)
{

/* If the EOF character has been reached then come out */
  if (feof(fpin_file))
    (break;)

/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if(row_num%EVERY_ROW == 0)
    (printf("Row = %d\n", row_num);)

/* Read in data band values, print them to output file */
  for(col=0; col<num_cols; col++)
  {
    fscanf(fpin_file,"%lf%*c", &input_value);
    fprintf(fpout_dn,"%10.6lf ",(float) input_value);
  }
  fprintf(fpout_dn,"\n");

/* Get ground data information, print it to output grnd data file */
  fgets(grnd_data, MAX_STRING_SIZE, fpin_file);
  fscanf(fpin_file, "\n");

  fputs(grnd_data,fpout_cl);

}/* End of for (;;) loop */

/* Free arrays and close files */
free(grnd_data);
fclose(fpin_file);
fclose(fpout_dn);
fclose(fpout_cl);

printf("\nPREPARE: 4 - Divide file vertically (e.g. into DN file and Class file...COMPLETE\n\n");

return;

} /* END of Divide_File subroutine */



/*########### SUBROUTINE: Change_Pixel_Codes ###########*/
void Change_Pixel_Codes(void)
{
/* Routine used to substitute codes with others or with ground
   data using a LU file */

  FILE   *fpin_file, *fpin_lu, *fpout_new;
  char   infile_name[MAX_FILENAME], inlu_name[MAX_FILENAME];
  char   outnew_name[MAX_FILENAME], class[MAX_STRING_SIZE];
  char   lu_class[MAX_STRING_SIZE], grnd_data[MAX_STRING_SIZE];
  char   thistime[MAX_STRING_SIZE], lasttime[MAX_STRING_SIZE];
  double input_value;
  int    col, num_cols, row_num=0;


/* Obtain names of files and information about structure */
fpin_file = My_Get_File("\nINPUT: Data file > ", infile_name, "r");

del_header(fpin_file);

printf ("\nNumber of columns ? > ");
scanf("%d%*c",&num_cols);

fpin_lu= My_Get_File("\nINPUT: Look up file > ", inlu_name, "r");
del_header(fpin_lu);

fpout_new= My_Get_File("\nOUTPUT: New file > ", outnew_name, "w");

printf("\nChanging the codes... \n");

/* For each row, until the end of the input file */
for (;;)
{

/* If the EOF character has been reached then come out */
   if (feof(fpin_file))
     (break;)

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    (printf("Row = %d\n", row_num);)


/* Read each input data source column and then the class,
   print the input data source to the new file */
  for(col = 0; col < num_cols - 1; col ++)
  {
    fscanf(fpin_file, "%lf%*c", &input_value);
    fprintf(fpout_new,"%8.6lf ",input_value);
  }
  fscanf(fpin_file,"%s%*c",class);


/* While old class in the input file and in the LU file
  are not equal and EOF in LU hasn't been reached, read
  the new information (class or ground data) */

  *lu_class = 'x' ;
  while (( strcmp(lu_class,class) != 0) && !feof(fpin_lu))
  {
    fscanf(fpin_lu,"%s%*c",lu_class);
    fgets(grnd_data,MAX_STRING_SIZE,fpin_lu);
  }


/* if you have reached the end of the lu file check whether
   it is because actually the old code didn't exist - only
   print error message once per class not found- otherwise
   print the new information to the new file */

  if feof(fpin_lu)
  {
    strcpy(thistime,class);

    if (strcmp(thistime,lasttime) !=0)
    {
      printf("\nThere was no new LU class for old class :");
      printf(" '%s' in the input file.\n",class);
    }

    fprintf(fpout_new, "%s\n", class);
    strcpy(lasttime,thistime);
  }
  else
    fputs(grnd_data,fpout_new);

  rewind(fpin_lu);

}/* End of do while not eof input file loop */


/* Free arrays and close files */
fclose(fpin_file);
fclose(fpin_lu);
fclose(fpout_new);

printf(" \nPREPARE : 6 - Change pixel codes using a Look Up Table...COMPLETE\n\n");

return;

} /* END of Change_Pixel_Codes subroutine*/




/*########### SUBROUTINE: Join_Cols ###########*/
void Join_Cols(void)
{
```



```
/* Routine used to join x columns of a first
   file with all the columns of a second file */

  FILE   *fpin_first, *fpin_second, *fpout_new;
  char   infirst_name[MAX_FILENAME], insecond_name[MAX_FILENAME],
         outnew_name[MAX_FILENAME];
  char   inputs_second[MAX_STRING_SIZE];
  double *inputs_first, input_value;
  int    col, num_cols, row, row_num=0;

/* Find out information and open files
        delheader used to skip any header lines */
fpin_first=My_Get_File("\nPlease input name of FIRST file > ", infirst_name, "r");

del_header(fpin_first);

printf("\nHow many columns from the first? > ");
scanf("%d%*c",&num_cols);


fpin_second=My_Get_File("\nPlease input name of SECOND file > ", insecond_name, "r");

del_header(fpin_second);

fpout_new=My_Get_File("\nPlease input name of OUTPUT joined file > ", outnew_name,"w");

/* Allocate memory */
inputs_first = dalloc(num_cols);


printf("\nJoining files ...\n");

/* Print header to output file */
fprintf(fpout_new,"#FIRST file : %s\n", infirst_name);
fprintf(fpout_new,"#SECOND file : %s\n\n", insecond_name);

for (;;)
{
/* If the EOF character has been reached then come out */
   if (feof(fpin_first)|| feof (fpin_second))
    (break;)

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    (printf("Row = %d\n", row_num);)

/* Read in the columns of data for the row of the
        first file and print them to the output file*/
  for(col=0;col<num_cols;col++)
  {
    fscanf(fpin_first,"%lf%*c", &input_value);
    fprintf(fpout_new,"%8.6lf ",input_value);
  }
/* If the EOF character has been reached then come out */
  if (feof(fpin_first))
    (break;)

/* Print some spaces and obtain and print the
   corresponding row of data from the second file */
  fprintf(fpout_new," ");
  fgets(inputs_second, MAX_STRING_SIZE,fpin_second);
  fputs(inputs_second, fpout_new);
} /* End of for loop */

/* Free arrays and close files */
free(inputs_second);
free(inputs_first);

fclose(fpin_first);
fclose(fpin_second);
fclose(fpout_new);

printf("\nPREPARE: 7 - Join files vertically...COMPLETE\n\n");


return;

} /* END of Join_Cols */



/*########### SUBROUTINE: Rm_Some_Codes ###########*/
void Rm_Some_Codes(void)
{
/* Routine used to remove pixels with a given code.
Used for example if a polygon is found to be
erroneously digitised or a class spectrally indistinct
*/

  FILE   *fpin_file, *fpin_lu, *fpout_new;
  char   infile_name[MAX_FILENAME], inlu_name[MAX_FILENAME],
         outnew_name[MAX_FILENAME];
  int    num_cols, class, lu_class, col, row_num;
  double *input_value;

/* Open files and obtain information
        del header used to ignore header lines if any */
fpin_file = My_Get_File("\nPlease input name of input data file > ",infile_name, "r");

del_header(fpin_file);

printf ("How many columns are there in this file > ");
scanf("%d%*c",&num_cols);

fpin_lu=My_Get_File("\nPlease input name of LOOK UP file > ", inlu_name, "r");
fpout_new=My_Get_File("\nPlease input name of NEW data file > ", outnew_name, "w");

/* Allocate memory */
input_value = dalloc(num_cols);

printf("\nRemoving...\n");

for (;;)
{
/* If the EOF character has been reached then come out */
   if (feof(fpin_file))
    (break;)

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    (printf("Row = %d\n", row_num);)

/* Read in all the non code information */
 for(col=0; col<num_cols - 1; col++)
    fscanf(fpin_file,"%lf%*c", &input_value[col]);

/* Read in the code */
  fscanf(fpin_file,"%d%*c", &class);


  lu_class = -1;

/* while the class read and the LU do not correspond, read down the
   LU file */

  while( (class != lu_class) && (!feof(fpin_lu)) )
    fscanf(fpin_lu,"%d%*c",&lu_class);

/* if class is not equal to a LU code then it wasn't in the LU file and
   therefore keep that row , otherwise discard it */

  if(class!=lu_class)
  {
    for(col = 0; col < num_cols-1; col ++)
      fprintf(fpout_new,"%8.6lf",input_value[col]);
    fprintf(fpout_new, " %d\n",class);
  }

  rewind(fpin_lu);
} /* End of for ;; loop */

/* Free arrays and close files */
free(input_value);

fclose(fpin_file);
fclose(fpin_lu);
fclose(fpout_new);

printf("\nPREPARE: 8 - Remove pixels that have the code given in the Look Up Table...COMPLETE\n\n");

return;

} /* END of Rm_Some_Codes subroutine */
```

```
/*############ SUBROUTINE: Check_Grnd ############*/

void Check_Grnd()
{
/* Routine used to check whether the given ground data
   percentages sum to 100% */

  FILE *fpin_tar;
  char intar_name[MAX_FILENAME], c;
  int  percentage, class, sum_per;
  int  i, count_row=0, row_num=0, ok=TRUE;


/* Obtain filenames and information */
fpin_tar = My_Get_File("\nINPUT: Ground data file > ", intar_name, "r");

del_header(fpin_tar);

printf("\nChecking...\n");

for (;;)
{

/* If the EOF character has been reached then come out */
   if (feof(fpin_tar))
     {break;}

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

  count_row += 1;

/* Read in the first pair percentage + class */
  fscanf(fpin_tar, "%d%*c%d%*c", &percentage, &class);

/* Read in all the percentages and calculate the sum for that pixel */
  sum_per = percentage;
  c = getc(fpin_tar);
  while(c != '\n')
  {
    fscanf(fpin_tar, "%d%*c%d%*c", &percentage, &class);
    c = getc(fpin_tar);
    sum_per = sum_per + percentage;
  }

/* Print warning message if sum was not 100 */
  if(sum_per != 100)
  {
    printf("\nWARNING: at row %d the total ground coverage was %d %%\n", count_row, sum_per);
    ok=FALSE;
  }
}

/* Print OK message */
if (ok == TRUE) printf("\nAll ground data vectors sum to 100%%.\n");

/* Close file */
fclose(fpin_tar);

printf("\nPREPARE: 9 - Check whether ground data percentages sum to 100%%...COMPLETE\n\n");

return;

} /* END of Check_Grnd */


/*############ SUBROUTINE: Sort_Grnd ############*/
void Sort_Grnd(void)
{
/* Routine used to sort ground data in increasing order
of percentge cover and to calculate composition matrix.
Composition matrix reports for each class the number of
times it occurs as a dominant, secondary,
third...class. If there are more than one percentage
the same then for both classes counted in the higest
positions. E.g. 40% 40% 20%, both classes with 40%
would be added 1 in their dominant class */

  FILE *fpin_tar, *fpout_tar, *fpout_comp;
  char intar_name[MAX_FILENAME], outtar_name[MAX_FILENAME];
  char outcomp_name[MAX_FILENAME];
  int *percentage, *class, *sor_per, c;
  int i, j,num_classes, row_num=0, old_per, new_per, equal=0, num_equal=0;
  int **count_class, *count_class_tot, *count_pos_tot, tot_tot=0, track = 0;
  int old_track = 0, x;


/* Obtain filenames and information and open files
        del header used to ignore any header lines */
fpin_tar = My_Get_File("\nINPUT: ground data filename > ", intar_name, "r");

del_header(fpin_tar);

printf("\nHow many 'pure' classes are there ? > ");
scanf("%d%*c", &num_classes);

fpout_tar = My_Get_File("\nOUTPUT: new ground data filename > ", outtar_name,"w");

fpout_comp = My_Get_File("\nOUTPUT: composition matrix > ", outcomp_name,"w");


/* Allocate memory and initialise arrays */
percentage = illoc(num_classes+1);
class = illoc(num_classes+1);
sor_per= illoc(num_classes+1);
count_class =int_matrix(num_classes+1, num_classes+1);
count_class_tot = illoc(num_classes+1);
count_pos_tot = illoc(num_classes+1);

for( i = 0; i <= num_classes; i++ )
{
  count_class_tot[i] = 0;
  count_pos_tot[i] = 0;
  for(j = 0; j<= num_classes; j ++)
    {count_class[i][j] = 0;}
}

printf("\nSorting & Calculating ... \n");


for (;;)
{

/* If the EOF character has been reached then come out */
   if (feof(fpin_tar))
     {break;}

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

/* Initialise arrays which much be initialised every
   row */
old_track = 0;
for( i = 0; i <= num_classes; i++ )
{
  percentage[i] = 0;
  class[i] = 0;
  sor_per[i] = 0;
}


/* Read in the information into a percentage array and
   a class array which correspond until an eol
        character has been reached */
  i = 1;
  fscanf(fpin_tar, "%d %d", &percentage[i], &class[i]);
  count_class_tot[class[i]] +=1;
  c=getc(fpin_tar);

/* If the EOF character has been reached then come out */
   if (feof(fpin_tar))
     {break;}

  while(c != '\n')
  {
    i+=1;
    fscanf(fpin_tar, "%d %d", &percentage[i], &class[i]);
    count_class_tot[class[i]] +=1;
    c = getc(fpin_tar);
  }
```



```
/* index the percentage array in increasing order of
   percentage from left to right */
  indexx(num_classes, percentage, sor_per);

/* While classes have 0 percentage, ignore them */
  i= 1;
  equal =0;
  while (percentage[sor_per[i]] == 0) {i = i+1;}


/* Keep track of the mixture components */
old_track = num_classes -i+1;
  if (track < old_track)
    {track = old_track;}

  old_per=percentage[sor_per[i]];

/*  Print the arrays in the sorted order */
  fprintf(fpout_tar, "%02d %02d ", percentage[sor_per[i]], class[sor_per[i]]);
  x=1;
  while (percentage[sor_per[i]] == percentage[sor_per[i+x]])
    {x=x+1;}

/* Count the occurence of each class in each position */
  count_class[class[sor_per[i]]][num_classes-i-(x-1)+1]+=1;

 for (j=i+1;j<=num_classes; j++)
 {
     fprintf(fpout_tar, "%02d %02d ", percentage[sor_per[j]], class[sor_per[j]]);
 }

  x=1;
  while (percentage[sor_per[j]] == percentage[sor_per[j+x]])
    {x=x+1;}

  count_class[class[sor_per[j]]][num_classes-j-(x-1)+1]+=1;

/* Account for equal percentages */
    new_per = percentage[sor_per[j]];
    if (new_per == old_per) {equal = 1;}
      old_per=new_per;
  }
 fprintf(fpout_tar,"\n");

/* Kepp track of whether there are pixels with equal
   percentages in their composition e.g. 40% 40% 20% */
if (equal==1) {num_equal = num_equal+1;}
} /* End of for (;;) loop */


/* Print the COMPOSITION matrix */
fprintf(fpout_comp, "#File used in composition matrix: %s\n", intar_name);


/* Print the header of the matrix */
fprintf(fpout_comp, "\n          COMPOSITION Matrix\n\n");
fprintf(fpout_comp, " Pos id-> \n");
fprintf(fpout_comp, "Class Num|");
for( i = 1; i <= track; i++)
  {fprintf(fpout_comp, "%6d", i);}
fprintf(fpout_comp, "  Class");
fprintf(fpout_comp, "\n----------|");
for( i = 0; i < track; i++ )
  {fprintf(fpout_comp, "------");}
fprintf(fpout_comp, "---");
fprintf(fpout_comp, "  Total");


/* For as many positions as the maximum number of mixture components
   print */
for(i = 1;i <= num_classes ; i++ )
{
  fprintf(fpout_comp, "\n%6d |", i);

 for (j=1; j<= track; j++)
    {fprintf(fpout_comp, "%6d", count_class[i][j]);}

  fprintf(fpout_comp, "   == %d%", count_class_tot[i]);
}
fprintf(fpout_comp, "\n");


for(i = 1;i <= num_classes ; i++ )
{
  for(j=1;j<=track;j++)
    {count_pos_tot[j] += count_class[i][j];}
}

for(i = 1;i <= num_classes ; i++ )
{tot_tot = count_class_tot[i] + tot_tot;}

fprintf(fpout_comp, "----------|");
for( i = 0; i < track; i++ )
  {fprintf(fpout_comp, "------");}
fprintf(fpout_comp, "---");

fprintf(fpout_comp, "\nPos Total|");

for(i = 1;i <= track ; i++ )
{ fprintf(fpout_comp, "%6d", count_pos_tot[i]);}

fprintf(fpout_comp, "   == %d\n", tot_tot);
fprintf(fpout_comp, "\n\nThere were %d pixels with equal coverages.\n", num_equal);


/* Free arrays and close files */
free(percentage);
free(class);
free(sor_per);
free(count_class_tot);
free(count_pos_tot);
free_i_matrix(count_class, num_classes+1);
fclose(fpin_tar);
fclose(fpout_tar);
fclose(fpout_comp);

printf("\nPREPARE: 10 - Sort ground data and produce statistics...COMPLETE\n\n");

return;

} /* END of Sort_Grnd */
```

```
/*****************************************************
* FILE: NNsoft_Analyse.c                             *
*                                                    *
* DESCRIPTION:                                       *
* Third main module of Soft Neural network program.  *
* Contains routines for the analysis of the output   *
* from the neural network. In particular, routines   *
* for indexing the outputs in ascending order,       *
* producing position and confusion matrices          *
* scaling, normalising, removing noise term, summing *
* the outputs and producing a 'bar' image that       *
* compares the ground data percentage coverage and   *
* that predicted by the neural network.              *
*                                                    *
* AUTHOR: A. C. Bernard                              *
*                                                    *
* LIST of routines: Create_Index_File                *
*                   Create_Rank_Matrices             *
*                   Create_Linear                    *
*                   Normalise                        *
*                   Sum_Max_Difference               *
*                   Make_Image                       *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_My_Additions.c: My_File_Open                *
*                        My_Get_File                 *
* NNsoft_utils.c       : illoc, dalloc...            *
*                                                    *
*                                                    *
* MODIFICATIONS :                                    *
*                                                    *
*****************************************************/


#include "NNsoft_neural.h"


/*############ SUBROUTINE: Create_Index_File ############*/
void Create_Index_File(void)
{
/* Routine used to create a file containing for each pixel,
   a vector of ground data classes ordered in increasing order
   of coverage of the pixel; a vector of neural network data
   classes ordered in increasing order of value; a vector which
   which indicates correct classification of a position by 1,
   incorrect classification by 2. 0 indicates positions of
   no relevance. For example, ordering of classes with 0% on
   the ground. Classes with the same percentages are accounted
   for */

  FILE   *fpin_nn, *fpin_grnd, *fpout_index;
  char   innn_name[MAX_FILENAME], ingrnd_name[MAX_FILENAME];
  char   outindex_name[MAX_FILENAME];
  float  *vector_nn, *target_grnd, min=0.0, min_nn=0.0;
  int    *nn_index_array, *grnd_index_array;
  int    num_classes, col, row_num=0;


/* Open files and obtain information
   del_header is used to ignore any header lines */
fpin_grnd = My_Get_File("\nINPUT: Ground data target/coverage file > ", ingrnd_name,"r");
del_header(fpin_grnd);

fpin_nn = My_Get_File("\nINPUT: Neural network outputs file > ", innn_name, "r");
del_header(fpin_nn);

printf("\nNumber of pure classes ? > ");
scanf("%d%*c",&num_classes);


/* The ranking of classes with minimum coverage (or neural network output
   value) e.g 0% for the ground (or e.g -1.7 for the neural network) is not
   of interest. Values below which data are not to be considered e.g 0% for
   ground data (or e.g. -1.6 for the neural network) are chosen here */
printf("\nValue in the ground target vector for the minimum? > ");
scanf("%f%*c",&min);

printf("\nValue in the neural network vector for the minimum? > ");
scanf("%f%*c",&min_nn);


fpout_index = My_Get_File ("\nOUTPUT: Index file > ", outindex_name, "w");


/* Set aside memory and initialise arrays */
vector_nn = falloc(num_classes+1);
target_grnd = falloc(num_classes+1);
nn_index_array = illoc(num_classes+1);
grnd_index_array = illoc(num_classes+1);

for( col = 0; col <= num_classes; col++ )
{
  vector_nn[col] = target_grnd[col] = 0.0;
  nn_index_array[col] = grnd_index_array[col] = 0;
}


/* Print header on index file */
fprintf(fpout_index, "#TARGET file : %s\n", ingrnd_name);
fprintf(fpout_index, "#NNET file  : %s\n", innn_name);
fprintf(fpout_index, "#Min for GRND: %3.1f\n", min);
fprintf(fpout_index, "#Min for NNet: %3.3f\n\n", min_nn);


printf("\n\nIndexing....\n");


/* while the eof character of the neural network file
   has not been reached...then do */
for(;;)
{

/* If the EOF character has been reached then come out */
  if (feof(fpin_nn) || feof(fpin_grnd))
    {break;}

/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

/* Read in the values of each class for the ground
   data target and the neural network outputs */
  for( col = 1; col <= num_classes; col++ )
  {
    fscanf(fpin_nn, "%f%*c",&vector_nn[col]);
    fscanf(fpin_grnd,"%f%*c",&target_grnd[col]);
  }

/* If the EOF character has been reached then come out */
  if (feof(fpin_nn) || feof(fpin_grnd))
    {break;}


/* Rank classes in ascending order of percentage coverage
   but do not modify arrays, instead  produce an index array */
   indexx(num_classes,vector_nn,nn_index_array);
   indexx(num_classes,target_grnd,grnd_index_array);


/* Print the ground index array. If the value in the ground target
   is less than or equal to the minimum specified above, e.g 0%
   coverage, then print 0 - we are not interested in the ranking of
   classes with minimum (e.g.0%) coverage */
  for( col = 1; col <= num_classes; col++ )
  {
    if (target_grnd[grnd_index_array[col]] > min)
      {fprintf(fpout_index, "%4d", grnd_index_array[col]);}
    else
      {fprintf(fpout_index, " 0");}
  }

  fprintf(fpout_index," ");


/* Print the neural network array. If the value in the neural network
   target is less than or equal to the minimum specified above, e.g -1.6
   then print 0 - we are not interested in the ranking of
   classes with minimum (e.g. -1.6) output values */
  for( col = 1; col <= num_classes; col++ )
  {
    if (vector_nn[nn_index_array[col]] > min_nn)
      {fprintf(fpout_index,"%4d",nn_index_array[col]);}
```



```
    else
      {fprintf(fpout_index, " 0");}
  }

  fprintf(fpout_index,"  ");


/* Compare index arrays. For classes where the ranking was set to 0
 comparison is 0. If classes of the same rank are the same, then
 print 1, if they are different, check that the ground cover of the
 calculated class is the same as the ground cover of the target
 class. If it is, then the ranks are the same print 1, if not, then
 they are different print 2
*/
  for( col = 1; col<= num_classes;col++)
  {
    if ((vector_nn[nn_index_array[col]] <= min_nn)|| (target_grnd[grnd_index_arr
ay[col]] <= min) )
      {fprintf(fpout_index, "0");}
    else if ( (grnd_index_array[col] == nn_index_array[col]) || (target_grnd[nn_
index_array[col]] == target_grnd[grnd_index_array[col]]) )
      {fprintf(fpout_index, "1");}
    else
      {fprintf(fpout_index, "2");}
  }

  fprintf(fpout_index, "\n");

} /* End of for (;;) loop */

/* Free arrays and close files */
free(vector_nn);
free(target_grnd);
free(nn_index_array);
free(grnd_index_array);

fclose(fpin_nn);
fclose(fpin_grnd);
fclose(fpout_index);

printf("\n\nANALYSE: 1 - Create index file...COMPLETE\n\n");

return;

} /* END of Create_Index_File subroutine */


/*############ SUBROUTINE: Create_Rank_Matrices ############*/
void Create_Rank_Matrices(void)
{
/* Routine used to create 2 matrices for analysis of the neural
   network outputs. The first 'Rank Matrix' analyses classes at
   every position of the array. It is like a confusion matrix in
   that it provides information as to where a class in position x
   has been classified. However it makes no differentiation between
   classes. The 2nd matrix analyses the arrays for misclassification.
   It looks at the class which the neural network gives for
   each position compared to the neural network. */

  FILE   *fpin_index, *fpout_matrix;
  char   inindex_name[MAX_FILENAME], outmatrix_name[MAX_FILENAME];
  char   del_line[MAX_STRING_SIZE];
  int    **pos_count, **miss_class, *pos_tot, *miss_tot, *count_true;
  int    *nn_index_array, *grnd_index_array, *vector_corr, prev_max=0, max=0;
  int    num_classes, col, row_num=0, j, x, all, i;
  double *per, *per_true, *miss_per;


/* Obtain filenames and information
   del_header is used to ignore any header lines */
fpin_index = My_Get_File("\nINPUT: Index file > ", inindex_name, "r");

del_header(fpin_index);

printf("\nNumber of pure classes ? > ");
scanf("%d%*c",&num_classes);

fpout_matrix = My_Get_File("\nOUTPUT: Matrices file > ", outmatrix_name, "w");


/* Set aside memory and initialise arrays */
pos_count = int_matrix(num_classes+1,num_classes+1);
miss_class = int_matrix(num_classes+1, num_classes+1);

nn_index_array = illoc(num_classes+1);
grnd_index_array = illoc(num_classes+1);
vector_corr = illoc(num_classes+1);
count_true = illoc(num_classes+1);
pos_tot = illoc(num_classes+1);
miss_tot = illoc(num_classes+1);

per = dalloc(num_classes+1);
miss_per = dalloc(num_classes+1);
per_true = dalloc(num_classes+1);

for( col = 0; col <= num_classes; col++ )
{
  for ( j = 0; j <= num_classes; j++)
    {pos_count[col][j] = miss_class[col][j] = 0;}
  nn_index_array[col] = grnd_index_array[col] = vector_corr[col]=0;
  count_true[col]=0; pos_tot[col] = miss_tot[col] = 0;
  per[col]= miss_per[col]= per_true[col]=0.0;
}

printf("\n\nCalculating...\n");


/* While the EOF character of the index file
   has not been reached...then do */
for (;;)
{

/* If the EOF character has been reached then come out */
  if (feof(fpin_index))
    {break;}

/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if(row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}


/* Read in the ground index array, the neural network index
   array and the array showing correctly/incorrectly classified
   classes. Since the arrays are in ascending order, read them
   in backwards for ease of comprehension of the matrix afterwards */
  for( col = 1; col <= num_classes; col++ )
    {fscanf(fpin_index, "%d%*c",&grnd_index_array[num_classes -col+1]);}


  for( col = 1; col <= num_classes; col++ )
    {fscanf(fpin_index, "%d%*c",&nn_index_array[num_classes-col+1]);}
  for( col = 1; col <= num_classes; col++ )
    {fscanf(fpin_index, "%d%*c",&vector_corr[num_classes-col+1]);}

/* If the EOF character has been reached then come out */
  if (feof(fpin_index))
    {break;}


/* Initialise variables which must be reinitialised for each
   pixel vector */
  col = 1;
  all = TRUE;
  j = 1;

/* While the ranking of the ground data is of interest (and
   therefore not indicated by 0) and there are still classes
   to compare, then produce the position matrix */
  while ( (col <= num_classes) && (vector_corr[col] != 0) )
  {

/* Find the position of the class given in the ground index array
   in the neural network index array while they differ.
   col is the position in the ground array, j is the position in
   the neural network array */
    j=1;
    while ((grnd_index_array[col] != nn_index_array[j]) && (j<=num_classes))
      {j=j+1;}


/* If the classes of the same rank are different but have the
```

```
   same coverage on the ground (vector_corr == 1) then the
   misclassification and rank matrices count the class as
   classified correctly otherwise the relevant incorrectly
   classified misclassification and rank counters are
   increased by 1 */
    if (vector_corr[col] == 1) /* and therefore rank is the same */
    {
      miss_class[grnd_index_array[col]][grnd_index_array[col]] +=1;
      pos_count[col][col] += 1;
    }
    else /* if(vector_corr[col] == 2) and therefore ranking is different */
    {
       miss_class[grnd_index_array[col]][nn_index_array[col]] +=1;
       all = FALSE;
       pos_count[col][j] += 1;
    }

/* If previous positions were correctly classified and true has therefore
   not been set to false, then increment the counter for truly correctly
   classified positions by 1. Truly correctly classified positions are those
   where previous positions are also correctly classified */
   if (all != FALSE)
    {count_true[col] += 1;}
   col += 1;

  } /* End of while (col<= num_classes) loop */

/* Keep a tag on the largest number of mixtures */
 max = col;
   if (max>prev_max) {prev_max = max;}
 count_true[1] = pos_count[1][1];


} /* End of for (;;) loop */

/* For each row in the matrices, calculate the sum of the row */
for(col=1; col<=num_classes; col++)
{
  for(j=1; j<=num_classes; j++)
  {
    pos_tot[col] = pos_tot[col] + pos_count[col][j];
    miss_tot[col] = miss_tot[col] + miss_class[col][j];
  }
}

/* For each possible position within a mixture (e.g. dominant, secondary etc)
   until the largest, calculate the percentage of the correctly classified
   pixels for that position and the percentage of truly correctly classified
   pixels */
for(col = 1; col < prev_max; col ++)
{
  per[col] = (double)pos_count[col][col]/(double)pos_tot[col]* 100.0;
  per_true[col] = (double)count_true[col]/(double)pos_tot[col]* 100.0;
}


/* For each class, calculate the percentage of correctly classifed
   pixels */
for(col = 1;col <= num_classes; col ++)
{
  if (miss_tot[col] != 0)
    {miss_per[col] = (double)miss_class[col][col]/(double)miss_tot[col]* 100.0;}
  else
    {miss_per[col] = 0.0;}
}


/* Print the rank confusion matrix and the modified misclassification
   matrix to a file. Print the header: name of the index file on which
   the matrices are based */
fprintf(fpout_matrix, "#File used in confusion matrix: %s\n", inindex_name);


/* Print the header of the Rank matrix */
fprintf(fpout_matrix, "\n      RANK confusion matrix\n\n");
fprintf(fpout_matrix, " pos_id|");
for( i = 1; i <= num_classes; i++)
  {fprintf(fpout_matrix, "%6d", i);}
fprintf(fpout_matrix, "  Total  - TRUE - Number of"); fprintf(fpout_matrix, "\n------
---|");
for( i = 0; i < num_classes; i++ )
  {fprintf(fpout_matrix, "------");}
fprintf(fpout_matrix, "  Accuracy  Accuracy  Pixels");


/* For as many positions as the maximum number of mixture components
   print the rank matrix values, the percentage correctly classifed
   per rank, the true percentage and the number of pixels which contain
   that many components */
for( col = 1; col <prev_max; col++ )
{
  fprintf(fpout_matrix, "\n%6d |", col);
  for (j=1; j<= num_classes; j++)
    {fprintf(fpout_matrix, "%6d", pos_count[col][j]);}

  fprintf(fpout_matrix, "  =%6.2f%%", per[col]);
  fprintf(fpout_matrix, "  -%6.2f%%",per_true[col]);
  fprintf(fpout_matrix, "  -(%d)",pos_tot[col]);
}
fprintf(fpout_matrix, "\n");


/* Print the header of the modified misclassification matrix */
fprintf(fpout_matrix, "\n\n\n      Modified MISCLASSIFICATION confusion matrix\n\n");
fprintf(fpout_matrix, "class_id|");
for( i = 1; i <= num_classes; i++ )
  {fprintf(fpout_matrix, "%6d", i);}
fprintf(fpout_matrix, "  Total  - Number of");
fprintf(fpout_matrix, "\n------|");
for( i = 0; i < num_classes; i++ )
  {fprintf(fpout_matrix, "------");}
fprintf(fpout_matrix, "  Accuracy  Pixels");


/* For all the classes, print the misclassification matrix and
   the percentage of correctly classified pixel positions */
for( col = 1; col <=num_classes; col++ )
{
  fprintf(fpout_matrix, "\n%6d |", col);
  for (j=1; j<= num_classes; j++)
    {fprintf(fpout_matrix, "%6d", miss_class[col][j]);}
  fprintf(fpout_matrix, "  =%6.2f%%", miss_per[col]);
  fprintf(fpout_matrix, "  -(%d)", miss_tot[col]);
}
fprintf(fpout_matrix, "\n");

/* Free arrays and close files */
free(count_true);
free(per);
free(miss_per);
free(per_true);
free(pos_tot);
free(miss_tot);
free(grnd_index_array);
free(nn_index_array);
free(vector_corr);
free_i_matrix(pos_count, num_classes);
free_i_matrix(miss_class, num_classes);

fclose(fpin_index);
fclose(fpout_matrix);

printf("\nANALYSE: 2 - Calculate rank matrices... COMPLETE\n\n");

return;

} /* END of Create_Rank_Matrices subroutine */


/*############ SUBROUTINE: Create_Linear ############*/
void Create_Linear(void)
{

/* Routine used to linearise the neural network outputs. The sigmoid
   function is applied to the neural network outputs before their
   value is produced. This routine inverts its effect a-posteriori so
   that neural network values are as if no sigmoid had been applied */

  FILE  *fpin_nn, *fpout_lin;
  char   innn_name[MAX_FILENAME], outlin_name[MAX_FILENAME];
  double nn_value, out_nn_lin;
  int    col, num_classes, row_num=0, inf;


/* Obtain filenames and information
   del_header is used to ignore any header lines */
fpin_nn = My_Get_File("\nINPUT: Neural network outputs filename > ", innn_name, "r");
```



```
del_header(fpin_nn);

fpout_lin = My_Get_File("\nOUTPUT: Linearised outputs filename > ", outlin_name, "w");
printf("\nNumber of classes ? > ");
scanf("%d%*c",&num_classes);


/* Print header to file */
fprintf(fpout_lin, "#NNET outputs file : %s\n\n", innn_name);

printf("\n Linearising... \n");


/* While the EOF character of the index file
   has not been reached...then do */
inf = FALSE;
for(;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_nn))
    {break;}


/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if(row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

  for(col=1; col<= num_classes; col++)
  {
    fscanf(fpin_nn, "%lf%*c", &nn_value);

/* Having read the neural network value which is a result of applying the
   activation function on the sum at that node, invert the effect of the
   activation function. activation function = tanh(), SCFACT = 1.7000;
   FACTOR = 1.3.
   IF nn_value = +/- 1.7, the inversion will lead to log 0 = Inf. Therefore
   set out_nn_lin to +/-7 as being a very high value compared to the
   others, but not infinite */
    if(nn_value == -SCFACT)
    {
      out_nn_lin = -7.0;
      inf = TRUE;
    }
    else if(nn_value == SCFACT)
    {
      out_nn_lin = 7.0;
      inf = TRUE;
    }
    else
      {out_nn_lin = (log(SCFACT+nn_value) -log(SCFACT-nn_value))/FACTOR;}

/* Print the new linearise value to the output file */
   fprintf(fpout_lin, "%3.6lf ",out_nn_lin);

  } /* End of for (col) loop */

  fprintf(fpout_lin, "\n");


}/* End of for (;;) loop */


/* If there was a neural network output value which equaled +/-1.7000 and
   which therefore causes the inversion algorithm to have to take the log
   of 0 which is infinite, print a warning message */
if (inf)
{
  printf("\nThere was at least one neural network output values = +/-1.7000");
  printf(" which meant that the linearised output was +/-Inf and therefore");
  printf(" set to +/-7.0. CHECK output file!\n");
}


/* Close files */
fclose(fpin_nn);
fclose(fpout_lin);

printf("\nANALYSE: 3 - Linearise NNet outputs...COMPLETE\n\n");

return;

} /* END of Create_Linear subroutine */


/*############ SUBROUTINE: Normalise ############*/
void Normalise(void)
{
/* Routine used to scale and/or normalise and/or remove a
   set value from each neural network output. Used prior
   to creating correspondance images */

  FILE   *fpout_scaled, *fpout_normed, *fpout_noise, *fpin_nn;
  char   outscaled_name[MAX_FILENAME], outnormed_name[MAX_FILENAME];
  char   innn_name[MAX_FILENAME], outnoise_name[MAX_FILENAME];
  char   ans_sca, ans_norm, ans_noise, ans_renorm, ans_all;
  double *scaled_val, *tmp_in, *noise_norm, *normed;
  double noise, max, oldmax, min, oldmin;
  double a,b, chk_sum, noise_sum;
  int    num_classes, col, row_num = 0, count_sum=0,count_noise=0;


/* Open files and obtain information */
fpin_nn = My_Get_File("\nINPUT: Neural network outputs file > ", innn_name, "r");

del_header(fpin_nn);


printf("\nNumber of classes ? : ");
scanf("%d%*c",&num_classes);

/* Find out whether outputs have already been scaled */
for (;;)
{
  printf("\nScale File ? (y/n) > ");
  scanf("%c%*c", &ans_sca);
  if (tolower(ans_sca) == 'y' || tolower(ans_sca) == 'n')
    {break;}
  printf("\ninvalid answer...");
}


/* If the input file does not contain scaled outputs,
   get information for the scaling */
if ( tolower(ans_sca) == 'y')
{
  fpout_scaled = My_Get_File("\nOUTPUT: Scaled file > ", outscaled_name, "w");
  printf("\nMin. value of scaled output ? > ");
  scanf("%lf", &min);
  printf("\nMax. value of scaled output ? > ");
  scanf("%lf%*c", &max);
  printf("\nOld Min. value ? > ");
  scanf("%lf%*c", &oldmin);
  printf("\nOld Max. value ? > ");
  scanf("%lf%*c", &oldmax);
  a = (min-max)/(oldmin-oldmax);
  b = ((oldmin*max)-(min*oldmax))/(oldmin-oldmax);

}


/* Find out whether to normalise the outputs */
for (;;)
{
  printf("\nNormalise output file ? (y/n) > ");
  scanf("%c%*c", &ans_norm);
  if ( tolower(ans_norm) == 'y' || tolower(ans_norm) == 'n')
    {break;}
  printf("Invalid answer...");
}


/* If the input file is to be normalised, find out
   name of output normalised file */
if ( tolower(ans_norm) == 'y')
  {fpout_normed = My_Get_File("\nOUTPUT: Normalised file > ", outnormed_name, "w");}


/* Find out whether to remove an amount from each output */
for(;;)
{
  printf("\n'Noise' normalise output file ? (y/n) > ");
  scanf("%c%*c", &ans_noise);
```

```
  if (tolower(ans_noise) == 'y' || tolower(ans_noise) == 'n' )
    {break;}
  printf("\nInvalid answer...");
}

/* If 'noise' is to be removed, find out how and how much */
if (tolower(ans_noise) == 'y')
{

/* Find out whether to remove the fixed amount from each output or
   whether to set outputs with a value <= to the fixed amount to
   0.00 but leave the other outputs untouched */
  for (;;)
  {
    printf("\nRemove from all ?(y/n) > ");
    scanf("%c%*c", &ans_all);
    if ( tolower(ans_all) == 'y' || tolower(ans_all) == 'n' )
      {break;}
    printf("Invalid answer...");
  }

  printf("\n'Noise' level ? > ");
  scanf("%lf%*c",&noise);


/* Find out whether to renormalise resulting outputs or not */
  for (;;)
  {
    printf("\nRe-normalise as well ? (y/n) > ");
    scanf("%c%*c", &ans_renorm);
    if (tolower(ans_renorm) == 'y' || tolower(ans_renorm) == 'n' )
      {break;}
    printf("Invalid answer...");
  }


/* Find out name of output file containing modified outputs */
  fpout_noise = My_Get_File("\nOUTPUT: 'Noise' treated file > ", outnoise_name, "w");

} /* End of if (noise) condition */
else /* no noise to be removed */
{
  ans_renorm = 'n';
  ans_all = 'n';
}



/* Print headers of files according to request */
  if (tolower(ans_sca) == 'y')
  {
    fprintf(fpout_scaled, "#Scaled file calculated from : %s \n", innn_name);
    fprintf(fpout_scaled, "#Old Min : %lf; Old Max. : %lf\n", oldmin, oldmax);
    fprintf(fpout_scaled, "#Min   : %lf; Max.   : %lf\n\n", min, max);
  }

  if (tolower(ans_norm) == 'y')
    {fprintf(fpout_normed, "#Normalised file calculated from : %s \n\n", innn_name);}

  if (tolower(ans_noise) == 'y')
  {
    fprintf(fpout_noise, "#'Noise' treated file calculated from : %s\n", innn_name);
    fprintf(fpout_noise, "#Noise level   : %f\n", noise);
    fprintf(fpout_noise, "#From all outputs : %c\n", ans_all);
    fprintf(fpout_noise, "#Renormalised   : %c\n\n", ans_renorm);
  }



/* Allocate memory and initialise arrays */
tmp_in = dalloc(num_classes);
scaled_val = dalloc(num_classes);
noise_norm = dalloc(num_classes);
normed = dalloc(num_classes);

printf("\nCalculating...\n");


/* For each pixel in the input file */
for(;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_nn))
    {break;}


/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if(row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

/* Initialise value which must be initialised at every pixel */
  chk_sum = 0.0;
  noise_sum = 0.0;
  for (col = 0; col < num_classes; col++)
  {
/* Read in the values */
    fscanf(fpin_nn, "%lf%*c", &tmp_in[col]);

/* If scaling is necessary, scale */
    if (tolower(ans_sca) == 'y')
    {
      scaled_val[col] = tmp_in[col]*a+b;
      fprintf(fpout_scaled, "%3.6f", scaled_val[col]);
    }
    else /* ans_sca == 'n' */
      {scaled_val[col] = tmp_in[col];}

    chk_sum += scaled_val[col];

/* If noise must be removed, remove noise - in which ever way
   was chosen  */
    if (tolower(ans_noise) == 'y')
    {
      if (tolower(ans_all) == 'y')
      {
        noise_norm[col] = scaled_val[col] - noise;
        if (noise_norm[col] < 0.00) noise_norm[col] = 0.00;
      }
      else /* ans_all = no */
      {
        if (scaled_val[col]<=noise)
          noise_norm[col] = 0.00;
        else
          noise_norm[col] = scaled_val[col];
      }

      noise_sum = noise_sum + noise_norm[col];
    } /* End of if (ans_noise) condition */

  } /* End of for (col) loop */

/* If either sums of outputs are 0 set them to 1 - Highly
   unlikely for scaled value, more likely for noise sum */
  if (chk_sum == 0.0)
    {chk_sum = 1.0;}

  if (noise_sum == 0.0)
    {noise_sum = 1.0;}



/* Print the scaled/normalised/noise values as requested */
  for (col = 0; col < num_classes; col++)
  {
    if (tolower(ans_norm) == 'y')
      fprintf(fpout_normed, "%3.6f", scaled_val[col]/chk_sum);
    if (tolower(ans_renorm) == 'y')
      fprintf(fpout_noise, "%3.6f", noise_norm[col]/noise_sum);
    else
      fprintf(fpout_noise, "%3.6f", noise_norm[col]);
  }

  if (tolower(ans_sca) == 'y')
    fprintf(fpout_scaled, "\n");

  if (tolower(ans_noise) == 'y')
    fprintf(fpout_noise, "\n");

  if (tolower(ans_norm) == 'y')
    fprintf(fpout_normed, "\n");

} /* End of for (;;) loop */
```



```
/* Free arrays and close files */
free(noise_norm);
free(scaled_val);
free(tmp_in);
free(normed);

if (tolower(ans_sca) == 'y')
  fclose(fpout_scaled);
if (tolower(ans_noise) == 'y')
  fclose(fpout_noise);
if (tolower(ans_norm) == 'y')
  fclose(fpout_normed);
fclose(fpin_nn);

printf("\nANALYSE: 4 - Scale and/or normalise and/or 'noise' treat NNet outputs...COMPLETE\n\n");

return;

} /* END of Normalise subroutine */



/*############ SUBROUTINE: Sum_Diff ############*/
void Sum_Diff(void)
{
/* Routine which identifies classes with the maximum and next maximum
   neural network values. Prints the sum of the neural network outputs
   and the difference between the two maximums for each pixel */

  FILE   *fpout_chk, *fpin_nn;
  char   outchk_name[MAX_FILENAME], innn_name[MAX_FILENAME];
  int    row_num=0, col, num_classes;
  double *tmp_in, max_one, max_two, prevmax, sum;


/* Open files and obtain information  */
fpin_nn = My_Get_File("\nINPUT: Neural network outputs file > ", innn_name, "r");

del_header(fpin_nn);

printf("\nNumber of classes ? : ");
scanf("%d%*c",&num_classes);


fpout_chk = My_Get_File("\nOUTPUT: Sum and/or differences file > ", outchk_name, "w");

/* Print header */
fprintf(fpout_chk, "#Sums calculated for NNet file : %s\n", innn_name);
fprintf(fpout_chk, "#Pixel Sum First Second Difference\n\n");


/* Allocate memory and initialise arrays */
tmp_in = dalloc(num_classes);

for (col=1; col<num_classes; col++)
  {tmp_in[col] = 0.0;}

printf("\nCalculating...\n");


/* while the eof character of the neural network file
   has not been reached...then do */
for(;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_nn))
    {break;}


/* Keep track of progress by printing the row
   number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if(row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}


/* Initialise the sum of the neural newtwork outputs for each new pixel.
   Read in the neural network output vector, sum its components */
  sum = 0.0;
  for (col = 0; col < num_classes; col++)
  {
    fscanf(fpin_nn, "%lf%*c", &tmp_in[col]);
    sum = sum + tmp_in[col];
  }


/* Determine which is the maximum of the neural network output vector
   and which is the second maximum */
  prevmax = tmp_in[0];
  max_two = min(prevmax,tmp_in[1]);
  for (col = 1; col < num_classes; col++)
  {
    max_one = max(prevmax, tmp_in[col]);
    if( max_one > prevmax )
    {
      max_two = prevmax;
      prevmax = max_one;
    }
    else
      {max_two = max(max_two, tmp_in[col]);}
  }

/* Print the pixel number, the sum of the vector, the first maximum,
   the second maximum and the difference between the last two to the
   output file */
fprintf(fpout_chk, "%4d%5.4lf%5.4lf%5.4lf%5.4lf\n", row_num, sum, max_one,max_two, max
_one-max_two);

}


/* Free arrays and close files */
free(tmp_in);
fclose(fpout_chk);
fclose(fpin_nn);

printf("\nANALYSE : 5 - Sum and/or differences between first two maximums...COMPLETE\n\n");

return;

} /*END of Sum_Diff subroutine*/



/*############ SUBROUTINE: Make_Image ############*/
void Make_Image(void)
{
/* Routine which produces a ground data / neural network
   correspondance image from a file consisting the
   vector of ground data (e.g. target file) and the vector
   of neural network outputs. File produced is a raw
   image */

 FILE   *fpin_dat, *fpout_new;
  char   indat_name[MAX_FILENAME], outnew_name[MAX_FILENAME];
  Byte   buffer_grnd[MAX_BUFFER_SIZE], buffer_nnet[MAX_BUFFER_SIZE];
  Byte   buffer_btwn[BTWN_SIZE];
  double *grnd, *nnet;
  int    thickness, pos, pos_start, pos_finish, num_classes;
  int    col, row_num=0, num_cols, i;


/* Open files and obtain information  */
fpin_dat = My_Get_File("\nINPUT: Image Data file > ", indat_name, "r");

del_header(fpin_dat);

printf("\nNumber of classes ? : ");
scanf("%d%*c",&num_classes);

fpout_new = My_Get_File("\nOUTPUT: Correspondance Image file > ", outnew_name, "wb");

printf("\nHow many columns to represent 100%% ? > ");
scanf("%d%*c", &num_cols);


/* Allocate memory and initialise arrays */
grnd = dalloc(num_classes);
nnet = dalloc(num_classes);
for (col = 0; col < num_classes; col ++)
  {grnd[col] = nnet[col] = 0.0;}

/* Buffer BTWN is the array seperating the ground data
   and neural network */
```

```
for ( col = 0; col < BTWN_SIZE; col ++)
  {buffer_btwn[col]= (Byte)0;}


for (col=0; col < MAX_BUFFER_SIZE; col ++)
  {buffer_grnd[col] = buffer_nnet[col]= (Byte)0;}



printf("\nMaking image...\n");

row_num = 0;
for (;;)
{
/* Keep track of progress */
  row_num += 1;
  if(row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

/* Read in arrays */
  for ( col = 0; col < num_classes; col ++)
    {fscanf(fpin_dat, "%lf%*c", &grnd[col]);}

  for (col = 0; col < num_classes; col ++)
    {fscanf(fpin_dat, "%lf%*c", &nnet[col]);}

/* If the EOF character has been reached then come out  */
  if (feof(fpin_dat))
    {break;}

/* Calculate positions of start and end for each class and
   assign the appropriate class to the buffer arrays */
pos_finish=0;
for (col=0; col < num_classes; col ++)
{
   pos_start = pos_finish;
   pos_finish += (int) (grnd[col]*num_cols);
   for( pos = pos_start; pos < pos_finish; pos++)
     buffer_grnd[pos] = (Byte)(col+1);
}

pos_finish=0;
for (col=0; col < num_classes; col ++)
{
   pos_start = pos_finish;
   pos_finish += (int) (nnet[col]*num_cols);
   for( pos = pos_start; pos < pos_finish; pos++)
     buffer_nnet[pos] = (Byte)(col+1);
}


for (col=0; col < BTWN_SIZE; col ++)
  {buffer_btwn[col] = (char)0;}

fwrite (buffer_grnd,sizeof(Byte), num_cols-1,fpout_new);
fwrite (buffer_btwn,sizeof(Byte), BTWN_SIZE, fpout_new);
fwrite (buffer_nnet,sizeof(Byte), num_cols-1, fpout_new);

fprintf(fpout_new, "\n");


} /*end of for (;;) loop */


/* Free arrays and close files */
free(grnd);
free(nnet);
free(buffer_btwn);
free(buffer_nnet);
free(buffer_grnd);

fclose(fpin_dat);
fclose(fpout_new);

printf("\nANALYSE: 6 - Make pixel Ref/NNet info correspondance images...COMPLETE\n\n");

return;

} /* END of Make_Image subroutine */
```

```
/*****************************************************
*                                                    *
* FILE: NNsoft_CreateTar.C                           *
*                                                    *
* DESCRIPTION:                                       *
* Routines called by the Create Tar Menu of Prepare  *
* Menu within NNsoft_Main. Contains different target *
* options, namely, scaled between a min and max      *
* (e.g. -1 and +1), from a pure integer (usual       *
* target min and max as e.g. above), from ground     *
* data but representation is pure (i.e. 60 3 40 2 is *
* represented as pure in class 3) and binned target  *
* where according to the percentage coverage get a   *
* value (e.g. 0 = -1, 1-30% = -0.4 etc ). Also has   *
* so called occurence target option where if a class *
* is present then it is signalled by a 1 and if not  *
* by 0, regardless of percentage                     *
*                                                    *
* AUTHOR: A. C. Bernard                              *
*                                                    *
* LIST of routines: Pure_Integer                     *
*                   Pure_Target                      *
*                   Scaled_Target                    *
*                   Binned_Target                    *
*                   Occ_Target                       *
*                   Rescale_Vector                   *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_MyAdditions.c: My_File_Open                 *
*                       My_Get_File                  *
* NNsoft_utils.c:       dalloc, illoc ...            *
*                                                    *
*                                                    *
* MODIFICATIONS:                                     *
*                                                    *
*****************************************************/

@include "NNsoft_neural.h"


/*############ SUBROUTINE: Pure_Integer ############*/
void Pure_Integer(void)
{
/* Routine used to create a target from the grnd data when the
   grnd data is provided as a pure class integer */

  FILE *fpin_tar, *fpout_tar;
  char intar_name[MAX_FILENAME], outtar_name[MAX_FILENAME];
  double max, min, *target;
  int i, j, num_classes, class_id, row_num=0;


/* Obtain filenames and information */
fpin_tar = My_Get_File("\nINPUT: Integer class number file > ", intar_name, "r");

del_header(fpin_tar);


printf("\nTotal number of pure classes ? : ");
scanf("%d*c",&num_classes);

printf("\nMax value i.e the value which ");
printf("is given to the class with 100%% > ");
scanf("%lf*c", &max);

printf("\nMin value i.e the value which ");
printf("is given to the class with 0%% > ");
scanf("%lf*c", &min);

fpout_tar = My_Get_File("\nOUTPUT: Target file > ", outtar_name,"w");

/* Allocate memory* */
target = dalloc(num_classes);

printf("\nMaking the target...\n");


for (;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_tar))
     {break;}

/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
     (printf("Row = %d\n", row_num);)

/* Initialise target vector for every pixel */
  for ( i = 0; i < num_classes; i++ )
     target[i] = min;

/* read in integer number, create target */
  fscanf(fpin_tar, "%d*c", &class_id);
  target[class_id-1] = max;
  for ( i = 0; i < num_classes; i++ )
     fprintf(fpout_tar, "%6.4lf ",target[i]);
  fprintf(fpout_tar, "\n");

} /* end of for (;;) loop */


/* Free arrays and close files */
free(target);
fclose(fpin_tar);
fclose(fpout_tar);

printf("\nCTARGET: 1 – Pure integer...COMPLETE\n\n");

return;

} /* END of Pure_Integer subroutine */



/*############ SUBROUTINE: Pure_Target ############*/
void Pure_Target(void)
{
/* routine which produces a target which ignores all information
   besides the dominant class in a ground data file */

  FILE *fpin_tar, *fpout_tar;
  char intar_name[MAX_FILENAME], outtar_name[MAX_FILENAME], c;
  double max, min, *target;
  int *percentage, *class, prevmax, maxout, node, row_num=0;
  int i, j, num_classes;


/* Obtain filenames and information */
fpin_tar = My_Get_File("\nINPUT: Ground data file > ", intar_name, "r");

del_header(fpin_tar);


printf("\nTotal number of pure classes ? : ");
scanf("%d*c",&num_classes);

printf("\nMax value i.e the value which ");
printf("is given to the class with 100%% > ");
scanf("%lf*c", &max);

printf("\nMin value i.e the value which ");
printf("is given to the class with 0%% > ");
scanf("%lf*c", &min);

fpout_tar = My_Get_File("\nOUTPUT: Target file > ", outtar_name,"w");

/* Allocate memory */
target = dalloc(num_classes+1);
percentage = illoc(num_classes+1);
class = illoc(num_classes+1);

printf("\nMaking the target...\n");


for (;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_tar))
     (break;)

/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
     (printf("Row = %d\n", row_num);)
```



```
/* Initialise arrays at each pixel */
 for( i = 0; i < num_classes; i++ )
  {
    class[i] = percentage[i] = 0;
    target[i] =  min;
  }

/* Read in the grnd data information whichis in the format
   percentage  class  percentage class etc */
  j=0;
  fscanf(fpin_tar, "%d*c %d*c", &percentage[j], &class[j]);
  c = getc(fpin_tar);
  while ((c != '\n') && !feof(fpin_tar) )
  {
    j+=1;
    fscanf(fpin_tar, "%d*c %d*c", &percentage[j], &class[j]);
    c = getc(fpin_tar);
  }

/* Determine the class with the dominant coverage */
  node = class[0];
  prevmax = percentage[0];
  for( i = 1; i < num_classes; i++ )
  {
    maxout = max(prevmax, percentage[i]);
    if( maxout > prevmax)
    {
     prevmax = maxout;
     node = class[i];
    }
  } /* End of for (i) loop */

  target[node-1] = max;

/* Print the target to the output file */
  for (i=0; i < num_classes; i++)
    fprintf(fpout_tar, "%6.4lf ",target[i]);
  fprintf(fpout_tar, "\n");

} /* End of for (;;) loop */

/* Free arrays and close file */
free(percentage);
free(target);
free(class);

fclose(fpin_tar);
fclose(fpout_tar);

printf("\nCREATE_TAR : 2 – Pure target...COMPLETE\n\n");

return;

} /* END of Pure_Target subroutine */



/*############ SUBROUTINE: Scaled_Target ############*/
void Scaled_Target(void)
{
/* Routine  which scales the grnd data between min and maximum values
   e.g -1 and +1 or 0 and +1 */

  FILE *fpin_tar, *fpout_tar;
  char intar_name[MAX_FILENAME], outtar_name[MAX_FILENAME], c;
  int num_classes, *percentage, *class, i, j, row_num=0;
  double max, min, *target;

/* Obtain filenames and information */
fpin_tar = My_Get_File("\nINPUT: Ground data file > ", intar_name, "r");

del_header(fpin_tar);


printf("\nTotal number of classes ? : ");
scanf("%d*c",&num_classes);

printf("\nMin value i.e the value which ");
printf("is given to the class with 0%% > ");
scanf("%lf*c", &min);

printf("\nMax value i.e the value which ");
printf("is given to the class with 100%% > ");
scanf("%lf*c", &max);

fpout_tar = My_Get_File("\nOUTPUT: Target file > ", outtar_name,"w");

/* Allocate memory */
target = dalloc(num_classes+1);
percentage = illoc(num_classes+1);
class = illoc(num_classes+1);

printf("\nMaking the target...\n");


for (;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_tar))
     {break;}

/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
     (printf("Row = %d\n", row_num);)

/* Initialise arrays at every pixel */
  for( i = 0; i < num_classes; i++ )
  {
    class[i] = percentage[i] = 0;
    target[i] =  min;
  }

/* Read in grnd data information */
  j=0;
  fscanf(fpin_tar, "%d*c %d*c", &percentage[j], &class[j]);
  c = getc(fpin_tar);
  while ((c != '\n') && !feof(fpin_tar) )
  {
    j+=1;
    fscanf(fpin_tar, "%d*c %d*c", &percentage[j], &class[j]);
    c = getc(fpin_tar);
  }

/* Calculate scaled equivalent of grnd data for each class */
  for( i = 0; i < num_classes; i++ )
    if(class[i] != 0)
      target[class[i]-1] = (double)percentage[i]/100.0*(max-min) + min;

/* Print target to output file */
  for (i=0; i < num_classes; i++)
    fprintf(fpout_tar, "%6.4lf ",target[i]);
  fprintf(fpout_tar, "\n");

} /* End of for (;;) loop */

/* Free arrays and close files */
free(percentage);
free(target);
free(class);

fclose(fpin_tar);
fclose(fpout_tar);

printf("\nTARGET: 3 – Scaled target...COMPLETE\n\n");

return;

} /* END of Scaled_Target subroutine */



/*############ SUBROUTINE: Binned_Target ############*/
void Binned_Target(void)
{
/* Routine used to produce targets where each class is placed in a
   'bin' on the basis of its coverage of the pixel */

  FILE *fpin_tar, *fpout_tar;
  char intar_name[MAX_FILENAME], outtar_name[MAX_FILENAME], c;
  double *bin_value, *target;
  int *bin_pos, num_bins, num_classes, *percentage, *class;
  int i,j, row_num=0;

/* Obtain filenames and information */
fpin_tar = My_Get_File("\nINPUT: Ground data file > ", intar_name, "r");
```

```
del_header(fpin_tar);


printf("\nTotal number of pure classes ? : ");
scanf("%d*c",&num_classes);

printf("\nNumber of bins > ");
scanf("%d*c", &num_bins);


/* Allocate memory */
bin_pos = illoc(num_bins);
bin_value = dalloc(num_bins);
target = dalloc(num_classes);
percentage = illoc(num_classes);
class = illoc(num_classes);

/* Find out the target value for each bin */
for (i = 0; i< num_bins; i++)
{
  printf("\nTarget value for bin no. %d > ", i+1);
  scanf("%lf*c", &bin_value[i]);
}

/* Find out the range for each bin */
printf("\n0%% <= x <= ?%% = %3.2f > ", bin_value[0]);
scanf ("%d*c", &bin_pos[0]);
for(i = 1; i < num_bins; i++)
{
  printf("\n%d%% < x <= ?%% = %3.2f > ", bin_pos[i-1], bin_value[i]);
  scanf ("%d*c", &bin_pos[i]);
}

fpout_tar = My_Get_File("\nOUTPUT: Target file > ", outtar_name,"w");


printf("\nMaking the target...\n");

for (;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_tar))
    {break;}

/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

/* Initialise arrays at every pixel */
  for( i = 0; i < num_classes; i++ )
  {
    class[i] = 0;
    percentage[i] = 0;
    target[i] = (double) bin_value[0];
  }


/* Read in ground data information */
  j=0;

  fscanf(fpin_tar, "%d*c%d*c", &percentage[j], &class[j]);
  c = getc(fpin_tar);
  while((c != '\n') && !feof(fpin_tar) )
  {
    j+=1;
    fscanf(fpin_tar, "%d*c%d*c", &percentage[j], &class[j]);
    c = getc(fpin_tar);
  }

  for (i=0; i < num_classes; i++)
  {
    j = 0;
    while(percentage[i] > bin_pos[j])
      j = j+1;
    target[class[i]-1] = bin_value[j];
  }

/* Print target to output file */
  for (i=0; i < num_classes; i++)
    fprintf(fpout_tar, "%6.4lf ",target[i]);
  fprintf(fpout_tar, "\n");

} /* End of for (;;) loop */

/* Free arrays and close files */
free(percentage);
free(target);
free(class);
free(bin_value);
free(bin_pos);

fclose(fpin_tar);
fclose(fpout_tar);

printf("\nTARGET: 4 - Binned target...COMPLETE\n\n");

return;

} /* END of Binned_Target subroutine */


/************ SUBROUTINE: Occ_Target ************/
void Occ_Target(void)
{
/* Routine which creates a target by assigning the maximum value (+1) to
   all classes which are present in the mixture, regardless of actual
   percentage cover, and minimum value (-1) to all classes which are
   not present in the mixture. In effect, this type of target signals the
   presence or not of a class */

  FILE *fpin_tar, *fpout_tar;
  char intar_name[MAX_FILENAME], outtar_name[MAX_FILENAME], c;
  double max, min, *target;
  int *percentage, *class, prevmax, maxout, node, row_num=0;
  int i, j, num_classes;


/* Obtain filenames and information */
fpin_tar = My_Get_File("\nINPUT: Ground data file > ", intar_name, "r");

del_header(fpin_tar);

printf("\nTotal number of pure classes ? : ");
scanf("%d*c",&num_classes);

printf("\nMin value is the value which ");
printf("is given to the class with 0%% > ");
scanf("%lf*c", &min);

printf("\nMax value is the value which ");
printf("is given to the class with 100%% > ");
scanf("%lf*c", &max);

fpout_tar = My_Get_File("\nOUTPUT: Target file > ", outtar_name,"w");

/* Allocate memory */
target = dalloc(num_classes+1);
percentage = illoc(num_classes+1);
class = illoc(num_classes+1);

printf("\nMaking the target...\n");

for (;;)
{
/* If the EOF character has been reached then come out */
   if (feof(fpin_tar))
     {break;}

/* Keep track of progress by printing the row
    number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

/* Initialise arrays */
  for( i = 0; i < num_classes; i++ )
  {
    class[i] = 0;
    percentage[i] = 0;
    target[i] =  min;
  }


/* Read in ground data information */
```



```
  j=0;
  fscanf(fpin_tar, "%d*c%d*c", &percentage[j], &class[j]);
  c = getc(fpin_tar);
  while ((c != '\n') && !feof(fpin_tar) )
  {
    j=j+1;
    fscanf(fpin_tar, "%d*c%d*c", &percentage[j], &class[j]);
    c = getc(fpin_tar);
  }


/* For any class which has a percentage larger than the minimum,
   assign max value to it */
  for (i=0; i<num_classes; i++)
    if (percentage[i] > min)
      target[class[i]-1] = max;

/* Print target to output file */
  for (i=0; i < num_classes; i++)
    fprintf(fpout_tar, "%6.4lf ",target[i]);
  fprintf(fpout_tar, "\n");

} /* End of for (;;) loop */

/* Free arrays and close files */
free(percentage);
free(target);
free(class);

fclose(fpin_tar);
fclose(fpout_tar);

printf("\nTARGET: 5 - Occurence target...COMPLETE\n\n");

return;

} /* END of Occ_Targec subroutine */


/************ SUBROUTINE: Rescale_Vector ************/
void Rescale_Vector(void)
{
/* Routine used to rescale a vector's values 1 by 1. i.e.
   with different mins and maxs for each. */

  FILE *fpin_tar, *fpout_tar;
  char intar_name[MAX_FILENAME], outtar_name[MAX_FILENAME], c;
  int num_classes, i, j, num_cols, row_num=0;
  double max, min, oldmin, oldmax, a, b, skip, *old_target, *new_target;

/* Obtain filenames and information */
fpin_tar = My_Get_File("\nINPUT: data file > ", intar_name, "r");

del_header(fpin_tar);

printf("\nNumber of columns to skip ? : ");
scanf("%d*c",&num_cols);


printf("\nTotal number of classes ? : ");
scanf("%d*c",&num_classes);

printf("\nMin value of new network target > ");
scanf("%lf*c", &min);

printf("\nMax value of new network target > ");
scanf("%lf*c", &max);

printf("\nOld Min value ? > ");
scanf("%lf*c", &oldmin);

printf("\nOld Max value ? > ");
scanf("%lf*c", &oldmax);

fpout_tar = My_Get_File("\nOUTPUT: Target file > ", outtar_name,"w");


/* Allocate memory */
old_target = dalloc(num_classes);
new_target = dalloc(num_classes);

/* Calculate scaling factors */
a = (min-max)/(oldmin-oldmax);
b = ((oldmin*max)-(min*oldmax))/(oldmin-oldmax);


printf("\nMaking the target...\n");

for (;;)
{
/* If the EOF character has been reached then come out */
  if (feof(fpin_tar))
    {break;}

/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}

/* Initialise target */
  for( i = 0; i < num_classes; i++ )
    new_target[i] =  min;


/* Read in values to be skipped and print them to the output file */
  for( i = 0; i < num_cols; i++ )
  {
    fscanf(fpin_tar, "%lf*c", &skip);
    fprintf(fpout_tar, "%6.3lf ",skip);
  }

/* Read in old target */
  for( i = 0; i < num_classes; i++ )
    fscanf(fpin_tar, "%lf*c", &old_target[i]);

/* Scale old target */
  for( i = 0; i < num_classes; i++ )
    new_target[i] = old_target[i] *a +b;

/* Print new target to file */
  for (i=0; i < num_classes; i++)
    fprintf(fpout_tar, "%6.4lf ",new_target[i]);
  fprintf(fpout_tar, "\n");

} /* End of for (;;) loop */

/* Free arrays and close file */
free(target);
fclose(fpin_tar);
fclose(fpout_tar);

printf("\nTARGET: 6 - Rescale vector...COMPLETE\n\n");

return;

}
```

NNsoft_MyAdditions.c 1/4

```
/*****************************************************
* FILE: NNsoft_MyAdditions.c                         *
*                                                    *
* DESCRIPTION:                                       *
* Contains routines that replace some of the         *
* original routines written for the neural network   *
* In addition, a few general utility routines        *
*                                                    *
* AUTHOR: A. C. Bernard                              *
*                                                    *
* LIST of routines:    My_Classify_Image             *
*                      My_Write_Result               *
*                      My_Init_Weights               *
*                      My_Get_File                   *
*                      My_File_Open                  *
*                      del_header                    *
*                      clloc                         *
*                      free_c_matrix                 *
*                      Compare_Int_Arrays            *
*                                                    *
* DEPENDENCIES:                                      *
* NNsoft_lipclassify.c : read_log                    *
* NNsoft_netarch.c     : archread                    *
*                      : read_weights                *
* NNsoft_neuralnet.c   : forward_pass                *
*                      : initialise                  *
*                                                    *
* MODIFICATIONS:                                     *
*                                                    *
*****************************************************/


#include "NNsoft_neural.h"


/*########## SUBROUTINE: My_Classify_Image ##########*/
void My_Classify_Image()
{

/* Routine used to classify an image which must be in Band
   Interleaved by line format. Includes the possibility
   of producing fraction images. Requires neural network
   module. Replaces the routine Classify_Image written by
   Ioannis Kanellopoulos in the neural network software */


  FILE   *fpin_image, *fpout_gis, **fpout_per, *fpout_stats;
  char   inimage_name[MAX_FILENAME], outgis_name[MAX_FILENAME];
  char   outper_name[MAX_FILENAME], root_name[MAX_FILENAME];
  char   outstats_name[MAX_FILENAME];
  char   ans, ans_norm,  ext[SMALL+3];
  Byte   *classified, *buffer;
  double **pixel_line, **pixel_line_norm, **pixel_line_sca, *sum_line;
  int    total_num_pixels, lines, columns, i, j, m;
  int    offset, prevmax, maxout, node, row, row_num=0,  *stats;


/* Open files and obtain information */
fpin_image = My_Get_File("\nINPUT: Image to be classified > ", inimage_name, "r");

printf("\nNumber of Rows ? > ");
scanf("%d%*c",&lines);

printf("\nNumber of Columns ? > ");
scanf("%d%*c",&columns);


/* Read architecture of network from weights file
   initialise network, read trained weights */
fclose(My_Get_File("\nINPUT: Weights file > ", weights_name, "r"));

archread(weights_name);
initialise();
read_weights();


/* Read the values from the training log file */
read_log();


/* Open Classified and Stats files */
fpout_gis = My_Get_File("\n\nOUTPUT: Classified image file > ", outgis_name,"w");

fpout_stats = My_Get_File("\nOUTPUT: Statistics filename > ", outstats_name, "w");


/* Find out whether fraction image should be created */
for (;;)
{
  printf("\nProduce percentage coverage files (1 per class)? (y/n) > ");
  scanf("%c%*c", &ans);

 if (tolower(ans) == 'y' || tolower(ans) == 'n')
    {break;}
  else
    {printf("Invalid option...\n");}

} /* End of for (;;) loop */


/* If fraction images are to be created then find out how,
   find out the root name, create the full name, open the files */
if (tolower(ans) == 'y')
{
  for (;;)
  {
    printf("\nNormalised or Scaled outputs ? (n/s) > ");
    scanf("%c%*c", &ans_norm);
    if (tolower(ans_norm) == 'n' || tolower(ans_norm) == 's')
      {break;}
    else
      {printf("Invalid option...\n");}

  } /* End of for (;;) loop */

/* Allocate memory for each fraction image */
  fpout_per =(FILE **)malloc(num_out_nodes*sizeof(FILE *));

/* Find out name to call each fraction image. Each image
   filename made up of root_name + _c + class number +.bil */
  printf("\nRoot filename for percentage files? > ");
  scanf("%s%*c",root_name);
  strcat(root_name, "_c");
  strcpy(outper_name, root_name);
  for ( m = 0; m < num_out_nodes; m++ )
  {
    sprintf(ext, "%d", m+1);
    strcpy(outper_name, root_name);
    strcat(outper_name, ext);
    strcat(outper_name, ".bil");

/* Open each fraction image */
    fpout_per[m] = My_File_Open(outper_name,"w");

  } /* End of for (m) loop */

} /* End of if (tolower(ans) == y ) condition */


/* Print the number of rows, cols, and total number of pixels to
   the screen */
total_num_pixels = lines*columns;
printf("\nImage Information: \nRows = %d, Cols = %d, Total Pixels = %d\n", lines, columns, total_n
um_pixels);
printf("Number of Bands = %d, Number of Classes = %d\n", num_in_nodes, num_out_nodes);

/* Set aside memory and initialise arrays */
stats = illoc(num_out_nodes);
pixel_line= double_matrix(num_out_nodes,columns);
pixel_line_norm= double_matrix(num_out_nodes,columns);
pixel_line_sca = double_matrix(num_out_nodes,columns);
sum_line = dalloc(columns);
classified= ulloc(columns);
for( m = 0; m < num_out_nodes; m++ )
{
  stats[m] = 0;
  for ( j = 0;  j < columns; j++ )
  {
    pixel_line[m][j] = 0.0;
    pixel_line_norm[m][j] = pixel_line_sca[m][j] = 0.0;
    sum_line[j] = 0.0;
    classified[j] = (Byte)0;

  }
} /* End of for (m) loop */

buffer = ulloc(columns*num_in_nodes);
```

NNsoft_MyAdditions.c 2/4

```
input  = double_matrix(columns, num_in_nodes);
for(j = 0; j < num_in_nodes*columns; j++)
  {buffer[j] = (Byte)0;}

for(j = 0; j < columns; j++)
  for(m=0; m<num_in_nodes; m++)
    {input[j][m] = 0.0;}


printf("\n\nClassifying...\n");


/* For each row of the image... */
for( row = 0; row < lines; row++ )
{
/* Keep track of progress by printing the row
number when it is a multiple of EVERY_ROW */
  row_num += 1;
  if (row_num%EVERY_ROW == 0)
    {printf("Row = %d\n", row_num);}


/* Read in the DN values of 1 line, all bands, to the buffer array*/
  fread((Byte *)buffer, sizeof(Byte), columns*num_in_nodes,fpin_image);


/* For each pixel in the line, for each band assign the DN to the
   input array, modify it */
  for ( j = 0; j < columns; j++ )
  {
    for( m = 0, offset = 0; m < num_in_nodes; m++ )
    {
      input[j][m] = (double) buffer[j+offset];
      offset += columns;
    }
    for( m = 0; m < num_in_nodes; m++ )
      {input[j][m] = ( input[j][m] - mean[m] ) / sdeviation[m];}

  } /* End of for (j) loop */


/* For each pixel in the line */
  for( j = 0; j < columns; j++ )
  {
/* Pass the pixel through the network */
    forward_pass(j);

/* Initialise all variables for each pixel.
   Note that the scaling assumes neural network outputs between
   -1.7 and +1.7 */
    prevmax = outputs[num_hidden_layers+1][0];
    node = m = 0;
    pixel_line[m][j]= (double)outputs[num_hidden_layers+1][m];
    pixel_line_sca[m][j] = pixel_line[m][j]/(2*SCFACT) + 0.5;
    sum_line[j] = pixel_line_sca[m][j];

/* For each next output node */
    for( m = 1; m < num_out_nodes; m++ )
    {
/* Determine the class with the maximum output */
      maxout = max(prevmax, outputs[num_hidden_layers+1][m]);
      if( maxout > prevmax )
      {
        prevmax = maxout;
        node = m;
      }

/* If fraction images are to be produced, calculate values */
      if (tolower(ans) == 'y')
      {
        pixel_line[m][j]= (double) outputs[num_hidden_layers+1][m];
        pixel_line_sca[m][j] = pixel_line[m][j]/(2*SCFACT) +0.5;
        sum_line[j] = sum_line[j] +  pixel_line_sca[m][j];
      }

    } /* End of for (m) loop */

/* If fraction images are to be normalised, calculate value */
    if (tolower(ans_norm) == 'n')
    {
      for( m = 0; m < num_out_nodes; m++ )
        {pixel_line_norm[m][j] = pixel_line_sca[m][j]*100/sum_line[j];}
    }

/* Add 1 to the counter of dominant class array */
    stats[node]++;

/* The value of the classified image for the pixel being considered
   is the node with the maximum neural network output value */
    classified[j] = (Byte) (node + 1);

  }/* End of for (j) loop */


 /* Write the dominant class of the lineof pixels to the output
    classified file */
  fwrite((Byte *)classified, sizeof(Byte), columns, fpout_gis);


/* If fraction images were to be produced write the value of the
   line to the respective files, scaled or normalised */
  if ( tolower(ans) == 'y' && tolower(ans_norm) == 'n' )
  {
    for( m = 0; m < num_out_nodes; m++ )
    {
      for( j = 0; j < columns; j++ )
        {fprintf(fpout_per[m], "%c", (char)pixel_line_norm[m][j]); }
    }
  }
  else if ( tolower(ans) == 'y' && tolower(ans_norm) == 's' )
  {
    for( m = 0; m < num_out_nodes; m++ )
    {
      for( j = 0; j < columns; j++ )
        {fprintf(fpout_per[m], "%c", (char)(pixel_line_sca[m][j]*100));}
    }
  } /* End of if condition */

} /* End of for row loop */


/* Print the number of pixels for each class given by the
   neural network */
for(i = 0; i < num_out_nodes; i++)
  fprintf(fpout_stats, "Class %2d had %6d pixels \( percentage of total %6.2f%%\n", i+1,stats[i],
stats[i]*100/(float) total_num_pixels);


/* Free arrays and close files */
fclose(fpin_image);
fclose(fpout_gis);
if(tolower(ans)=='y')
{
  for(m=0; m<num_out_nodes; m++)
    {fclose(fpout_per[m]);}
}
fclose(fpout_stats);

free_d_matrix(fpout_per, num_out_nodes);

free(buffer);
free(classified);
free(mean);
free(sdeviation);
free_d_matrix(input,columns);
free_d_matrix(pixel_line, num_out_nodes);
free_d_matrix(pixel_line_norm, num_out_nodes);
free(sum_line);

printf("\n\nNEURAL NETWORK: 3 - Classification...COMPLETE\n\n");

return;

} /* END of My_Classify_Image subroutine */


/*########## SUBROUTINE: My_Write_Result ##########*/
void My_Write_Result(int npatterns, char *outvec_name)
{
/* Routine used to write out a file of neural network
   outputs at the testing stage of the neural network.
   Called in NNsoft_lipclassify.c */

  FILE   *fpout_vec;
  int    row, col;
```

```
/* Open the output file provided as a parameter */
fpout_vec = My_File_Open(outvec_name, "a");


/* For every input pattern in the test file ..*/
for( row = 0; row < npatterns; row++ )
{

/* Pass the pattern through the neural network. */
  forward_pass(row);

/* Print each output value to the output file */
  for ( col = 0; col < num_out_nodes; col++ )
    {fprintf(fpout_vec, "%3.5f ", outputs[num_hidden_layers + 1][col]);}

  fprintf(fpout_vec,"\n");

} /* End of for (row) loop */

/* Close files */
fclose(fpout_vec);

return;

} /* END of My_Write_Result */



/*########### SUBROUTINE: My_Init_Weights ###########*/
void My_Init_Weights()
{
/* Function that uses the current time
to seed the initial weights - consequently, every
time the net is run, the weights have a different
initial value. The function ran3 which is part of
numerical receipes in C can be replaced by any other
random number generator which, like ran3, produces values
between 0 and 1. In this way weights are scaled between
-0.5 and +0.5. Routine called by NNsoft_neural_net.c*/

  int i, j;
  int idum;

/* Seed idum with the current time */
idum = time(NULL)*-1;

/* For as many hidden layers and as many hidden nodes in each
   layer - so in other words for was many weights as there are,
   initialise weights, deltas and gradient */
for( j = 0; j < num_hidden_layers + 1; j++ )
  for( i = 0; i < (num_hidden_nodes[j+1] * num_hidden_nodes[j+1]; i++ )
  {
    *(weights[j] + i) = (double) (ran3(&idum) - 0.5);
    *(deltas[j] + i) = 0.0;
    *(gradient[j]+i) = 0.0;
  }

return;

} /* END of My_Init_Weights subroutine */



/*########### SUBROUTINE: My_Get_File ###########*/
FILE *My_Get_File(char *prompt,char *filename, char *mode)
{
  FILE *fp;

/* Loop until a correct filename has been entered */
for (;;)
{

/* Ask for filename and read answer */
  printf("%s",prompt);
  scanf("%s%*c", filename);

/* If the filename exits (fopen does not return NULL),
   open the file, otherwise print that cannot find file */
  if ( (fp = fopen (filename,mode)) != NULL)
    {break;}

  printf("Cannot find file '%s'\n", filename);
}

return fp;

} /* END of My_Get_File subroutine */



/*########### SUBROUTINE: My_File_Open ###########*/
FILE *My_File_Open(char filename[MAX_FILENAME], char mode[SMALL])
{
/* Modified from Ioannis' file_open */

  FILE *fp;

/* Loop until a correct filename has been entered */
while ( (fp = fopen (filename, mode) )== NULL)
{
  printf("Cannot find file '%s'\n", filename);
  printf("\nINPUT filename again > ");
  scanf("%s%*c", filename);

}

return fp;

} /* END of My_File_Open subroutine */



/*########### SUBROUTINE: del_header ###########*/
void del_header (FILE *fp)
{
/* Routine used for reading and ignoring X header
   lines in a file. Used in most routines */

  char del_line[MAX_STRING_SIZE];
  int num_heads, row;

/* Find out how many lines to skip */
printf("\nNumber of header lines to skip ? > ");
scanf("%d%*c", &num_heads);

/* Skip the lines */
for( row = 0; row < num_heads; row++ )
  fgets(del_line,MAX_STRING_SIZE,fp);

return;

} /* End of del_header subroutine */



/*########### SUBROUTINE: clloc ########### */
char *clloc(int size)
{
/* Routine written on the model of those in
   NNsoft_utils.c written by Ioannis Kanellopoulos
   and used allocate memory for a character matrix */

  char *v;

if( ( v = (char *) malloc(sizeof(char) * size ) ) == NULL )
{
  printf("\nError: Memory allocation failed in clloc().");
  exit(1);
}

return v;

} /* END of clloc */



/*########### SUBROUTINE: free_c_matrix ###########*/
void free_c_matrix(char **v,int vsize)
{
/* Routine written on the model of those in
   NNsoft_utils.c written by Ioannis Kanellopoulos
   and used to free a character matrix */

  int i;

for(i = 0; i < vsize; i++)
  {free((char *) v[i]);}
```



```
free((char *) v);

return;

} /* End of free_c_matrix subroutine */



/*########### SUBROUTINE: Compare_Int_Arrays ###########*/
int Compare_Int_Arrays(int *first_array, int *second_array, int num_columns)
{
/* Routine which is not used anywhere but kept for
   eventual use. Used to compare two integer arrays. If
   they are equal returns 1 else returns 0 */

 int col=0, are_equal=TRUE;

/* Compare arrays element by element. As soon as a difference
   is encountered, loop ends */
while (are_equal && (col < num_columns))
{
  if (first_array[col] != second_array[col])
    {are_equal = FALSE;}

  col ++;
}

return(are_equal);

} /* END of Compare_Int_Arrays subroutine */
```

Appendix B

REFERENCE DATA INFORMATION

B.1 Sample Survey Sheet for the Portugal Site

COMMISSION OF THE EUROPEAN COMMUNITIES - JOINT RESEARCH CENTRE
CORINE LAND COVER UPDATING PROJECT - LISBON FIELD EXPERIMENT 1991

B *FORM B: HOMOGENEOUS LAND COVER PARCEL DESCRIPTION (Min. 4ha*

TEST SITES:
TS1: Lagoa de Albufeira
TS2: Setubal
TS3: Vila Fresca de Azeitao
TS4: Gambia
TS5: Belem
TS6: Loures
TS7: Alenquer
TS8: Samora Correia A
TS9: Poceirao
TS10: Coruche
TS11: Samora Correia B

TEST SITE:- [T][S][ ]

LAND COVER PARCEL NUMBER:- [ ][ ][ ]
[should be shown on air-photos & maps with boundary]

DATE SURVEYED:- [ ][ ]-[J][U][N]-[9][1] TIME:- [ ][ ]:[ ][ ]

FILM/PHOTO:- [ ]/[ ] SURVEYORS' INITIALS:- 1. _____ 2. _____

LAND COVER TYPE:- _____
(Please be specific and brief)

GENERAL CONDITIONS:- _____

VEGETATION DETAILS:-

	HEIGHT	%COVER	DOMINANT SPECIES
LAYER 1:			
LAYER 2:			
LAYER 3:			

SURFACE DETAILS (Where not covered by vegetation):-

PREDOMINANT COLOUR: _____

WETNESS: VERY WET / WET / DRY (Circle which one applies)

COMPOSITION: (AREA %)

WATER	SOIL (circle one): Type? Clayey / Sandy	STONES	ROCK	CONCRETE
[ ][ ][ ]	[ ][ ][ ]	[ ][ ][ ]	[ ][ ][ ]	[ ][ ][ ]

TILE	METAL	ASPHALT	OTHER (Name:_____)
[ ][ ][ ]	[ ][ ][ ]	[ ][ ][ ]	[ ][ ][ ]

ADDITIONAL COMMENTS:-

Please continue on back of form if necessary.....

B.2 Quantile-Quantile Plots for the 'Portugal 16 Class' Data Set

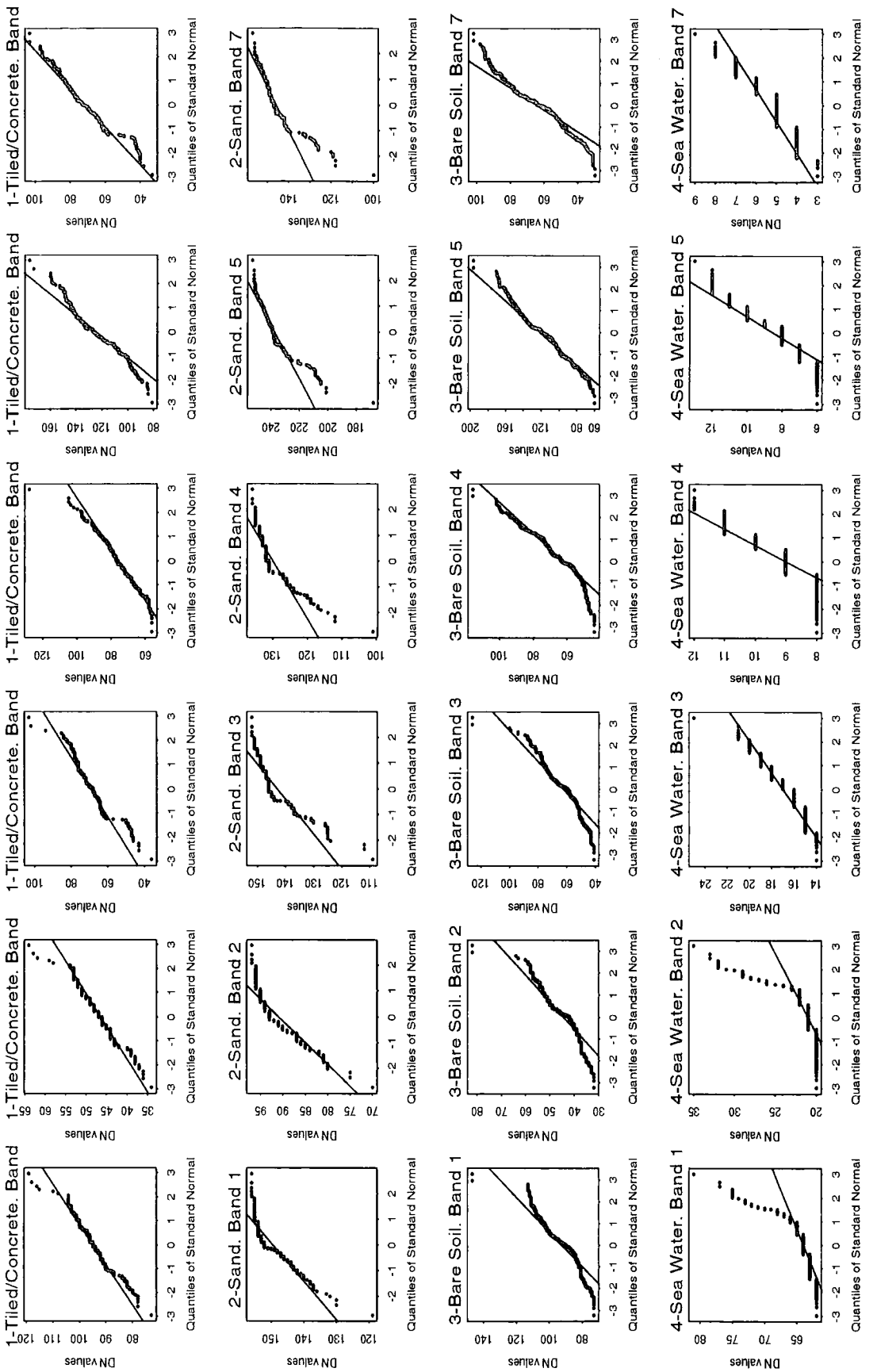


Figure B-1. Quantile-quantile plots for all the bands for classes 1-4 of the 'Portugal sixteen class' data set

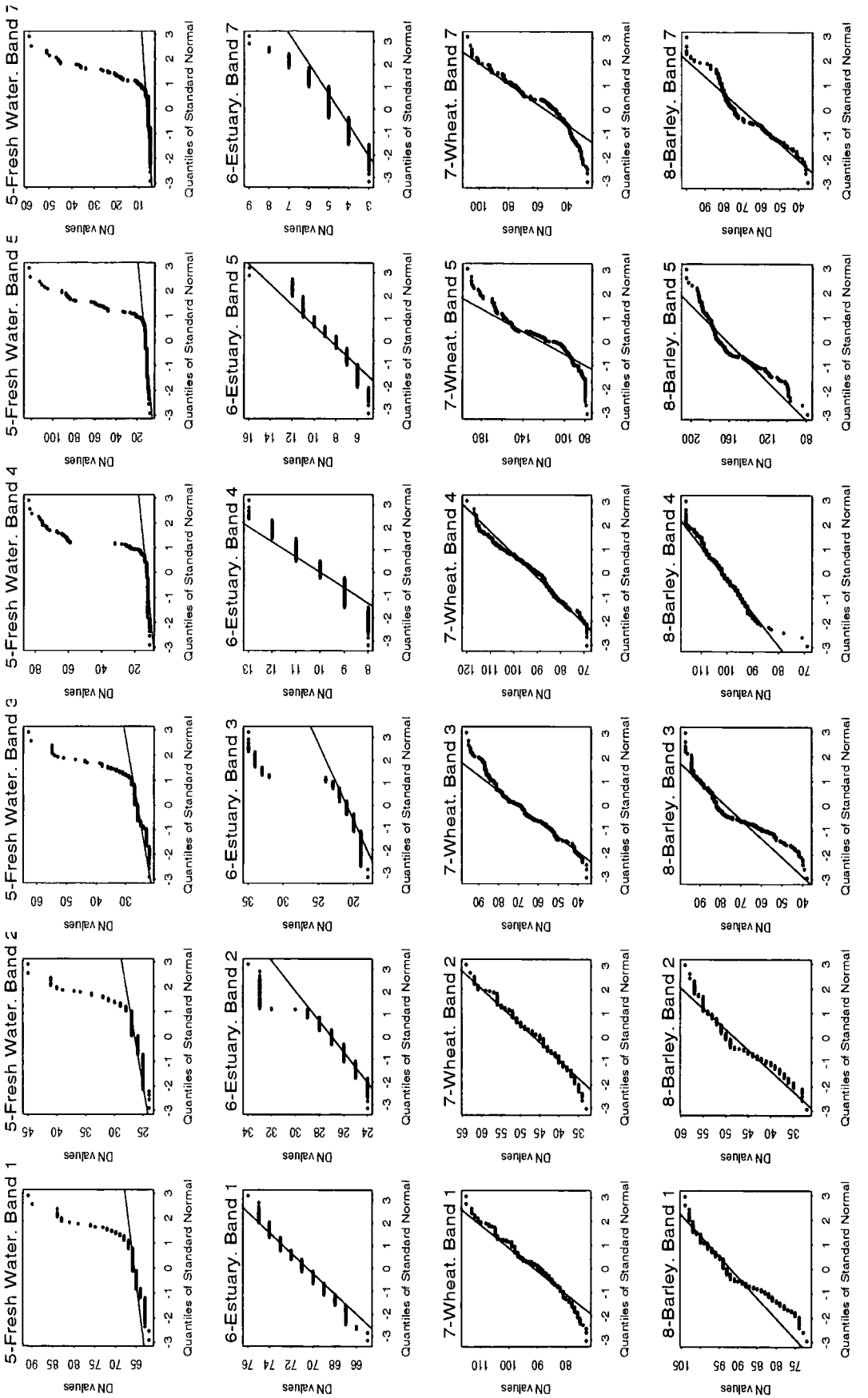


Figure B-2. Quantile-quantile plots for all the bands for classes 5 to 8 of the 'Portugal sixteen class' data set

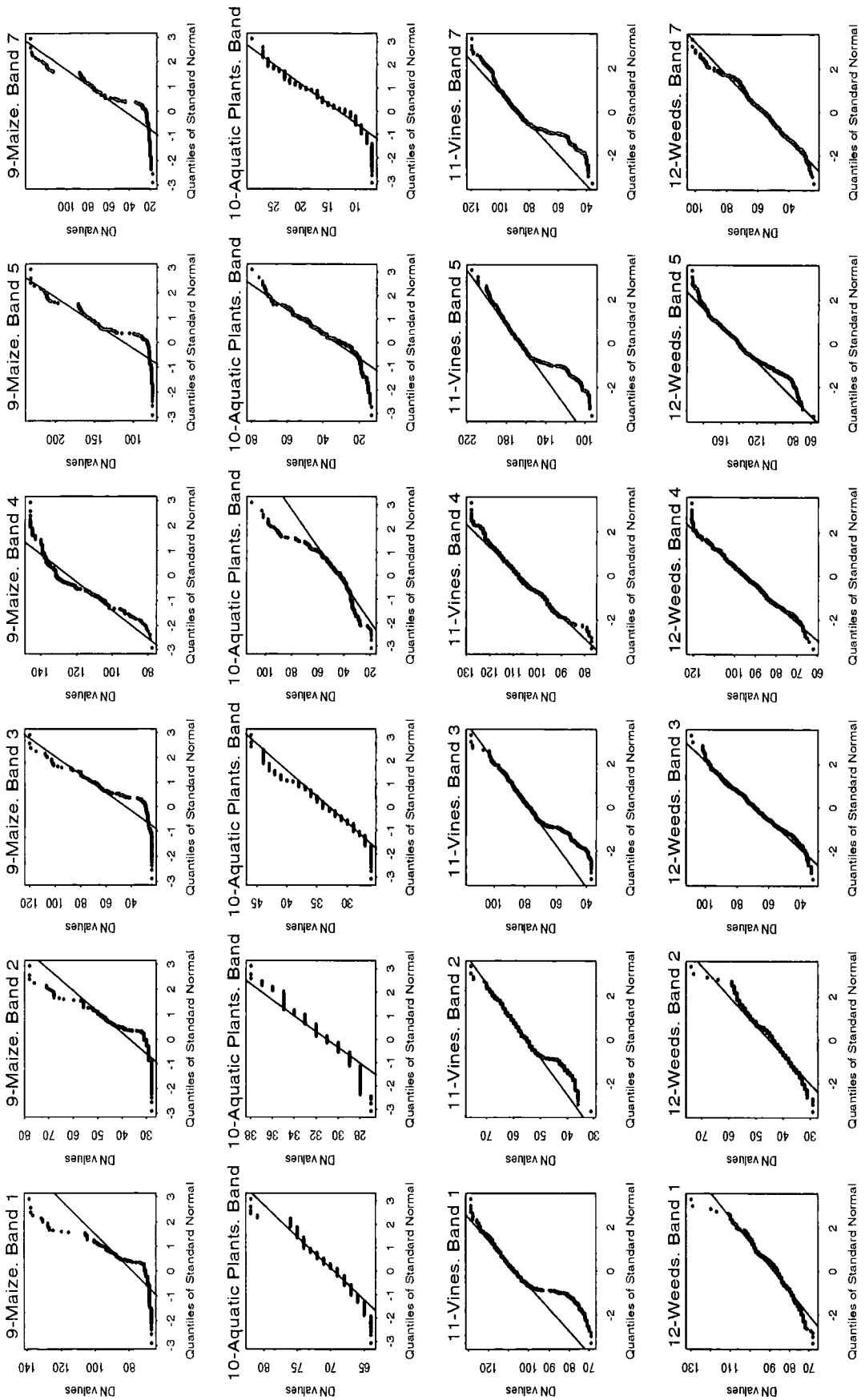


Figure B-3. Quantile-quantile plots for all the bands for classes 9 to 12 of the 'Portugal sixteen class' data set

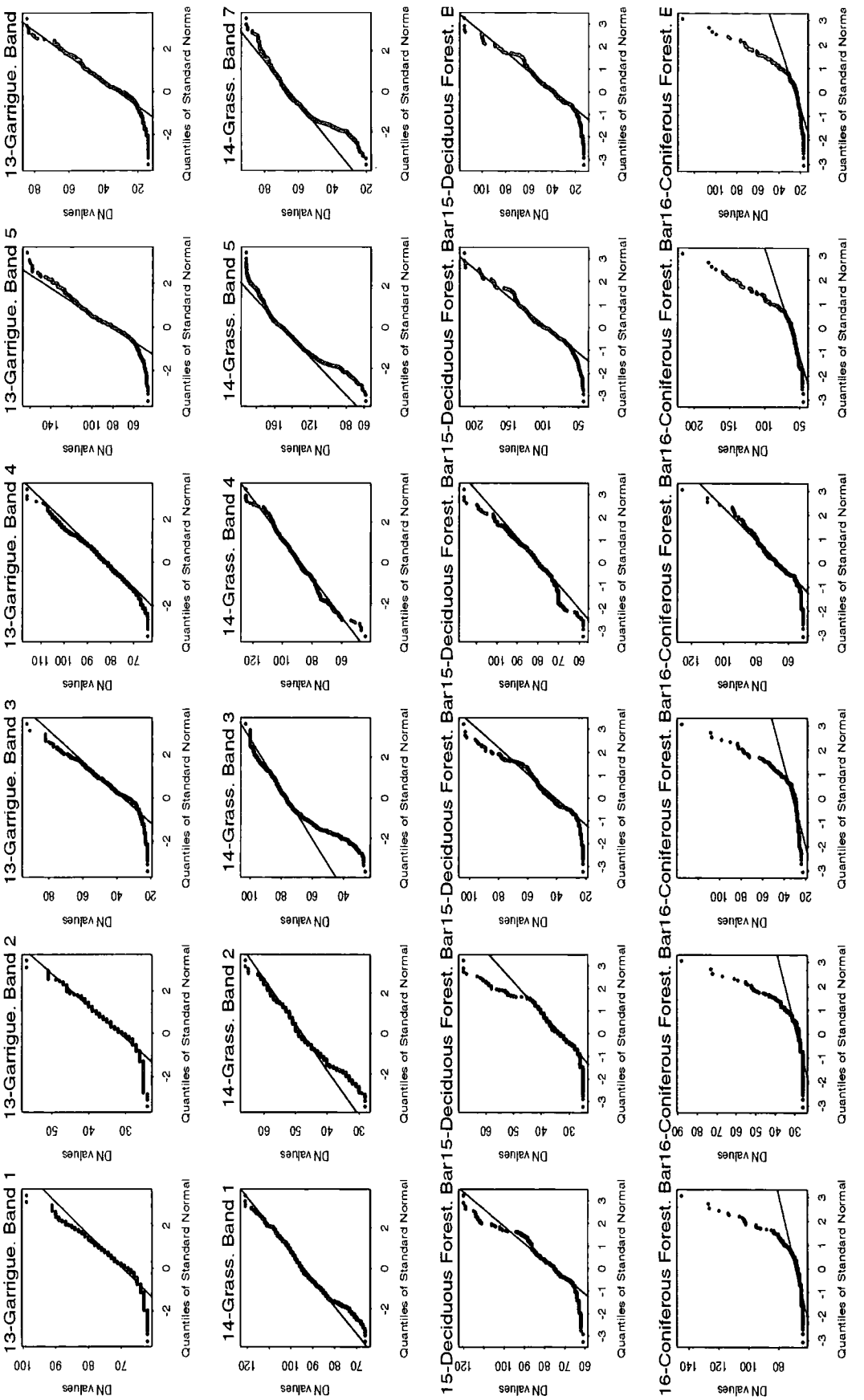


Figure B-4. Quantile-quantile plots for all the bands for classes 13 to 16 of the 'Portugal sixteen class' data set

A.3 List of Mixtures in the 'Portugal Fifteen Class' Data Set

Mixture types are to be read as follows: data are in pairs, the first integer represents the percentage cover of a polygon by the second integer which is the class. The number of mixture components is indicated by the number of pairs on a line. Percentage cover must add to 100%.

Mixtures present in the 'Portugal fifteen class' data set
100 01
100 02
100 04
100 06
100 07
100 08
100 10
20 01 80 04
10 01 90 05
60 01 40 06
80 01 20 06
40 01 10 06 50 09
50 01 10 06 40 10
20 01 80 07
30 01 70 07
40 01 60 07
10 01 90 08
30 01 70 08
80 01 20 08
30 01 70 09
40 01 60 09
60 01 40 09
40 01 50 09 10 10
10 01 90 10
20 01 80 10
30 01 70 10
40 01 60 10
70 01 30 10
80 01 20 10
20 01 50 10 30 11
20 01 60 10 20 11
20 01 70 10 10 11
30 01 10 10 60 11
30 01 40 10 30 11
50 01 20 10 30 11
50 01 30 10 20 11
40 01 30 10 30 11
60 01 10 10 30 11
60 01 20 10 20 11
80 01 10 10 10 11
30 01 20 10 40 11 10 13
30 01 50 10 10 11 10 13
30 01 30 10 10 12 30 13
10 01 40 10 50 13
10 01 50 10 40 13
20 01 30 10 50 13
60 01 10 10 30 13
10 01 20 10 70 14
10 01 40 10 50 14
10 01 60 10 30 14
10 01 70 10 20 14
20 01 40 10 40 14
20 01 50 10 30 14
50 01 30 10 20 14
70 01 10 10 20 14
20 01 20 10 50 14 10 15
20 01 40 10 20 14 20 15
20 01 40 10 30 14 10 15
30 01 30 10 20 14 20 15
10 01 20 10 70 15
20 01 40 10 40 15
40 01 60 11
60 01 40 11
70 01 30 11
80 01 20 11
90 01 10 11
40 01 50 11 10 13
50 01 20 11 30 13
50 01 30 11 20 13
60 01 20 12 20 13
10 01 30 12 60 14
20 01 60 12 20 14

20 01 30 12 20 14 30 15
30 01 50 12 10 14 10 15
10 01 30 12 60 15
20 01 70 12 10 15
60 01 10 12 30 15
50 01 50 13
10 03 90 12
10 04 90 05
20 04 80 05
20 04 80 10
50 05 50 10
60 06 40 10
40 06 60 12
60 06 40 12
50 08 50 10
90 08 10 14
10 10 90 12
90 10 10 12
20 10 40 12 40 14
30 10 50 12 20 15
50 10 40 12 10 15
70 10 30 13
80 10 20 13
10 10 90 14
40 10 60 14

60 10 40 14
30 10 70 15
40 12 60 14
20 12 30 14 50 15
20 12 50 14 30 15
10 12 90 15
20 12 80 15
30 12 70 15
40 12 60 15
60 12 40 15
80 12 20 15
Total Number of Mixtures = 108

A.4 List of Mixtures Removed to Form the 'Portugal Fifteen Class' Data Set

Mixtures not included in the 'Portugal fifteen class' data set
02 01 98 07
25 01 75 07
50 01 50 07
05 01 95 08
65 01 35 08
85 01 15 08
95 01 05 08
50 01 50 09
75 01 25 09
85 01 15 09
05 01 95 10
50 01 50 10
75 01 25 10
99 01 01 10
15 01 85 11
50 01 50 11
75 01 25 11
85 01 15 11
95 01 05 11
97 01 03 11
98 01 02 11
05 01 95 15
95 04 05 10
05 06 95 10
02 03 98 12
95 08 05 13
99 10 01 12
75 10 25 13
25 10 75 14
65 10 35 14
75 10 25 14
98 10 02 14
05 10 95 15
02 12 98 15
06 12 94 15
65 12 35 15
85 12 15 15
99 12 01 15
40 01 02 03 58 11
05 01 05 03 90 12
08 01 02 03 90 12
15 01 15 03 70 12
15 01 15 04 70 05
50 01 25 06 25 09
78 01 20 06 02 09
38 01 60 08 02 10
90 01 05 08 05 14
93 01 02 09 05 10
60 01 25 09 15 13
05 01 85 10 10 11
20 01 65 10 15 11
20 01 75 10 05 11
65 01 05 10 30 11
65 01 20 10 15 11
70 01 05 10 25 11
75 01 10 10 15 11
90 01 01 10 09 11
90 01 05 10 05 11
02 01 90 10 08 12
02 01 97 10 01 12
09 01 90 10 01 12
15 01 80 10 05 12
03 01 69 10 28 13
08 01 90 10 02 13
45 01 05 10 50 13
75 01 05 10 20 13
05 01 35 10 60 14
50 01 48 10 02 15
50 01 48 11 02 13
65 01 30 11 05 13
70 01 25 11 05 13
80 01 15 11 05 13
01 01 95 12 04 13
01 01 60 12 39 14
02 01 18 12 80 14
02 01 63 12 35 14
01 01 49 12 50 15
01 01 98 12 01 15
02 01 08 12 90 15
02 01 40 12 58 15
02 01 58 12 40 15
02 01 94 12 04 15
02 01 97 12 01 15
18 01 80 12 02 15
01 01 40 14 59 15
15 03 05 06 80 12
02 03 70 10 28 12
05 03 05 10 90 15
01 03 90 12 09 15
50 06 45 10 05 12
04 06 81 10 15 13
05 06 05 10 90 15
98 10 01 12 01 14
06 01 90 03 03 06 01 10
10 01 03 03 83 12 04 15
47 01 02 03 04 06 47 09
02 01 02 06 18 12 78 15
05 01 65 06 15 12 15 15
20 01 15 06 60 10 05 13
40 01 05 06 50 10 05 13
50 01 15 08 20 09 15 13
05 01 65 10 28 11 02 13
20 01 50 10 25 11 05 13
35 01 30 10 30 11 05 13
35 01 50 10 10 11 05 13
40 01 05 10 50 11 05 13
40 01 25 10 15 11 20 13
40 01 50 10 05 11 05 13
75 01 05 10 15 11 05 13
80 01 10 10 05 11 05 13

05 01	15 10	40 12	40 14
05 01	54 10	40 12	01 15
10 01	40 10	25 14	25 15
20 01	10 10	35 14	35 15
05 01	45 12	35 14	15 15
03 03	02 06	80 12	15 15
04 03	06 06	70 12	20 15
02 06	70 10	05 12	23 14
10 01	10 03	15 06	35 10	30 13
45 01	03 03	02 06	45 10	05 15
45 01	05 03	25 12	20 14	05 15
02 01	02 06	35 10	05 12	56 14
02 01	03 06	25 10	40 12	30 13
05 03	60 06	10 10	15 12	10 15
Total Number of Mixtures = 125				

Mixtures removed when buffering
100 03
60 01 40 05
10 01 90 07
20 01 80 08
50 01 50 08
70 01 30 08
70 01 30 09
80 01 20 09
90 01 10 09
97 01 03 09
02 01 98 10
60 01 40 10
90 01 10 10
95 01 05 10
98 01 02 10
99 01 01 11
20 01 80 13
90 01 10 13
80 06 20 10
85 10 15 11
40 10 60 15

80 12	20 13		
50 12	50 15		
90 12	10 15		
45 01	05 06	50 09	
10 01	60 07	30 10	
40 01	50 07	10 13	
50 01	20 08	30 13	
60 01	20 08	20 10	
15 01	70 09	15 10	
20 01	35 09	45 10	
60 01	20 09	20 10	
80 01	15 09	05 10	
30 01	50 09	20 13	
40 01	10 09	50 13	
90 01	05 09	05 13	
45 01	50 10	05 11	
50 01	40 10	10 11	
80 01	05 10	15 11	
05 01	60 10	35 13	
10 01	60 10	30 13	
15 01	80 10	05 13	
20 01	20 10	60 13	
30 01	10 10	60 13	
30 01	20 10	50 13	
40 01	30 10	30 13	
40 01	40 10	20 13	
50 01	30 10	20 13	
70 01	05 10	25 13	
10 01	80 10	10 14	
75 01	05 10	20 14	
10 01	70 10	20 15	
75 01	05 12	20 14	
04 01	95 12	01 15	
15 06	80 10	05 12	
10 06	80 10	10 14	
15 01	02 03	73 12	10 15
01 01	24 06	60 10	15 13
74 01	01 08	05 09	20 13
65 01	10 10	20 11	05 13
03 01	75 10	02 12	20 13

05 01	50 10	20 12	25 15
10 01	86 10	02 12	02 15
45 01	20 10	15 12	20 13
10 01	20 10	35 14	35 15
60 01	05 11	20 12	15 15
40 01	20 12	20 14	20 15
15 01	02 03	01 06	81 10	01 13
15 01	02 03	02 06	76 10	05 13
50 01	05 06	25 10	10 11	10 13
Total Number of Mixtures = 70				

B.5 Composition Matrices for the 'Portugal 15 Class' Data Sets

#File used in composition matrix: Portugal_15_Dataset

COMPOSITION Matrix

Pos id -> / Class Num	1	2	3	4	Class Total
1	5485	8119	4234	0	== 17838
2	102	0	0	0	== 102
3	0	133	0	0	== 133
4	1292	356	0	0	== 1648
5	208	0	0	0	== 208
6	713	281	181	0	== 1175
7	938	0	0	0	== 938
8	672	5	0	0	== 677
9	129	6	0	0	== 135
10	11493	3641	2238	0	== 17372
11	2195	3970	72	0	== 6237
12	1640	2199	565	52	== 4456
13	63	135	72	129	== 399
14	1914	4564	765	0	== 7243
15	2141	1281	764	131	== 4317
Pos Total	28985	24690	8891	312	== 62878

There were 5256 pixels with equal coverages.

#File used in composition matrix: Portugal_15_Training

COMPOSITION Matrix

Pos id -> / Class Num	1	2	3	4	Class Total
1	1779	2739	1419	0	== 5937
2	33	0	0	0	== 33
3	0	40	0	0	== 40
4	428	151	0	0	== 579
5	91	0	0	0	== 91
6	237	98	57	0	== 392
7	321	0	0	0	== 321
8	231	1	0	0	== 232
9	45	2	0	0	== 47
10	3865	1175	724	0	== 5764
11	712	1302	24	0	== 2038
12	551	725	187	20	== 1483
13	22	39	23	36	== 120
14	647	1496	252	0	== 2395
15	718	415	253	35	== 1421
Pos Total	9680	8183	2939	91	== 20893

There were 1734 pixels with equal coverages.

#File used in composition matrix: Portugal_15_Testing

COMPOSITION Matrix

Pos id -> / Class Num	1	2	3	4	Class Total
1	3706	5380	2815	0	== 11901
2	69	0	0	0	== 69
3	0	93	0	0	== 93
4	864	205	0	0	== 1069
5	117	0	0	0	== 117
6	476	183	124	0	== 783
7	617	0	0	0	== 617
8	441	4	0	0	== 445
9	84	4	0	0	== 88
10	7628	2466	1514	0	== 11608
11	1483	2668	48	0	== 4199
12	1089	1474	378	32	== 2973
13	41	96	49	93	== 279
14	1267	3068	513	0	== 4848
15	1423	866	511	96	== 2896
Pos Total	19305	16507	5952	221	== 41985

There were 3522 pixels with equal coverages.

B.6 Composition Matrices for the 'Portugal 7 Class' Data Sets

#File used in composition matrix: Portugal_7_Dataset

COMPOSITION Matrix

Pos id -> / Class Num	1	2	Class Total
1	1968	1536	== 3504
2	102	0	== 102
3	1196	162	== 1358
4	528	0	== 528
5	372	0	== 372
6	4525	3121	== 7646
7	3161	0	== 3161
Pos Total	11852	4819	== 16671

There were 0 pixels with equal coverages.

#File used in composition matrix: Portugal_7_Training

COMPOSITION Matrix

Pos id -> / Class Num	1	2	Class Total
1	1286	1023	== 2309
2	69	0	== 69
3	799	113	== 912
4	363	0	== 363
5	253	0	== 253
6	3018	2068	== 5086
7	2113	0	== 2113
Pos Total	7901	3204	== 11105

There were 0 pixels with equal coverages.

#File used in composition matrix: Portugal_7_Testing

COMPOSITION Matrix

Pos id -> / Class Num	1	2	Class Total
1	682	513	== 1195
2	33	0	== 33
3	397	49	== 446
4	165	0	== 165
5	119	0	== 119
6	1507	1053	== 2560
7	1048	0	== 1048
Pos Total	3951	1615	== 5566

There were 0 pixels with equal coverages.

B.7 Sample Survey Sheet for the Scotland Site

MAY
~~APRIL~~ 11-13,1996

(12)

DATE	12/05/96	SLOPE	0°
MAP-ID	5/6	ASPECT	—
PHOTO-ID		XCOORD	53793
POLY-ID	75	YCOORD	81131
CMPT #	710	PLOT #	1

NX GPS 53760 81246

TREE #	CIRCUM.	HT ANG. FOOT	HT. ANG TOP	SPECIES
1	10			
2	12			
3	12			
4a	11			
b	7			
5	9			
6	13			
7	10			
8	5			
9	11	-7	33	
10a	7			
b	5			
11	13			
12a	12			
b	6			
13	14			
14	6			

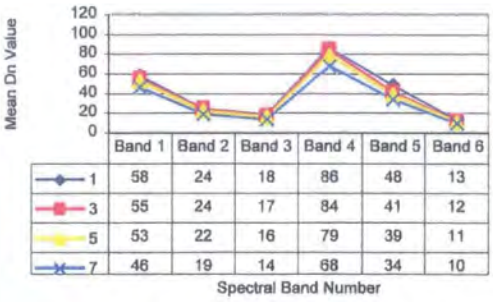
COMMENTS:

Pure SS
Open canopy
Grass understory with moss

NB Open ground within plot

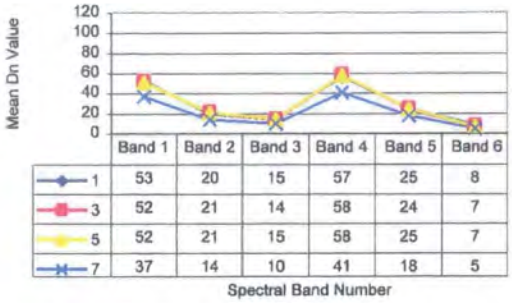
B.8 Variograms for 'Scotland Bio All' and 'Scotland Bio Box' Data

Plot 24



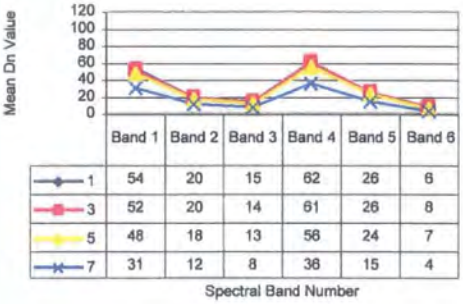
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	58	24	18	86	48	13
3	55	24	17	84	41	12
5	53	22	16	79	39	11
7	46	19	14	68	34	10

Plot 101



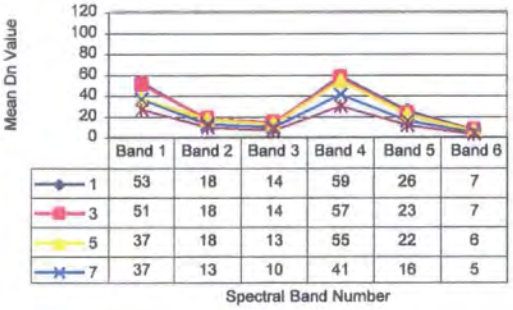
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	53	20	15	57	25	8
3	52	21	14	58	24	7
5	52	21	15	58	25	7
7	37	14	10	41	18	5

Plot 109



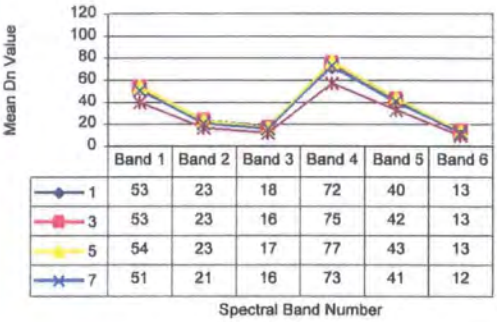
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	54	20	15	62	26	8
3	52	20	14	61	26	8
5	48	18	13	56	24	7
7	31	12	8	36	15	4

Plot 114



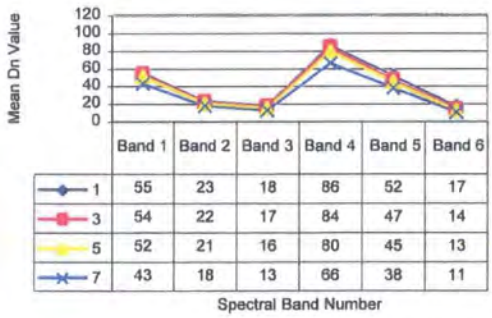
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	53	18	14	59	26	7
3	51	18	14	57	23	7
5	37	18	13	55	22	6
7	37	13	10	41	16	5

Plot 41



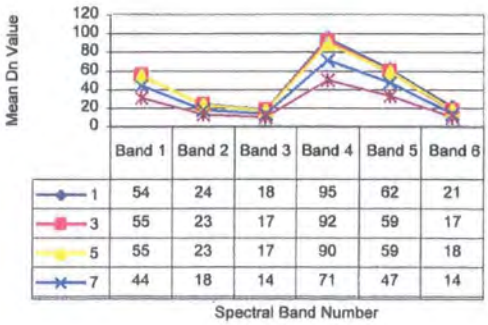
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	53	23	18	72	40	13
3	53	23	16	75	42	13
5	54	23	17	77	43	13
7	51	21	16	73	41	12

Plot 40



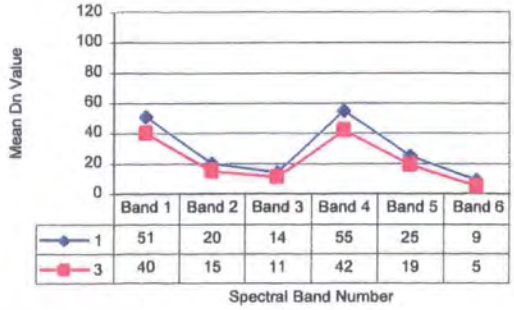
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	55	23	18	86	52	17
3	54	22	17	84	47	14
5	52	21	16	80	45	13
7	43	18	13	66	38	11

Plot 39



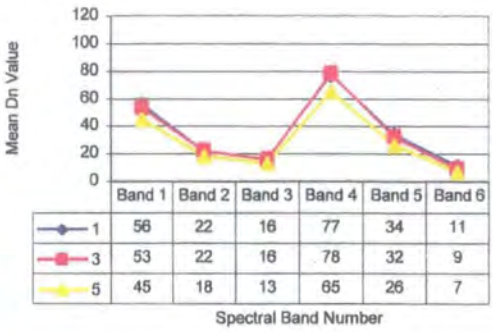
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	54	24	18	95	62	21
3	55	23	17	92	59	17
5	55	23	17	90	59	18
7	44	18	14	71	47	14

Plot 2



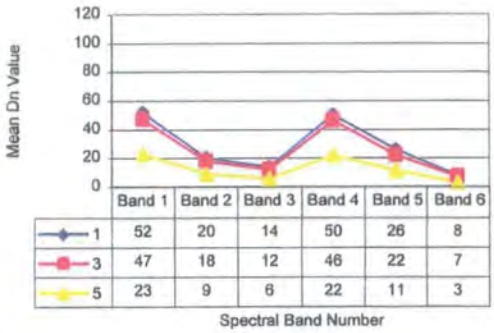
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	51	20	14	55	25	9
3	40	15	11	42	19	5

Plot 54



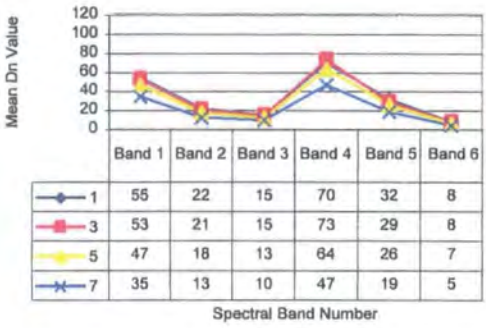
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	56	22	16	77	34	11
3	53	22	16	78	32	9
5	45	18	13	65	26	7

Plot 49



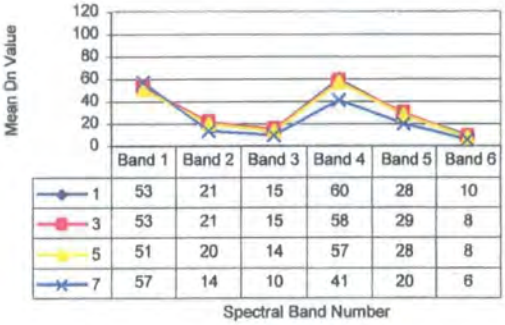
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	52	20	14	50	26	8
3	47	18	12	46	22	7
5	23	9	6	22	11	3

Plot 72



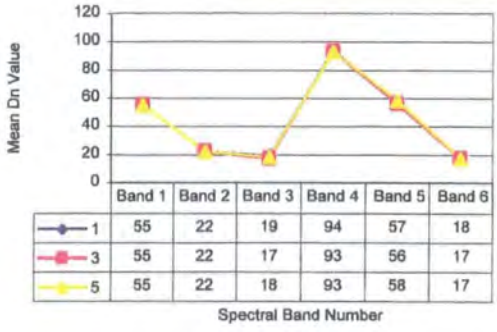
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	55	22	15	70	32	8
3	53	21	15	73	29	8
5	47	18	13	64	26	7
7	35	13	10	47	19	5

Plot 59



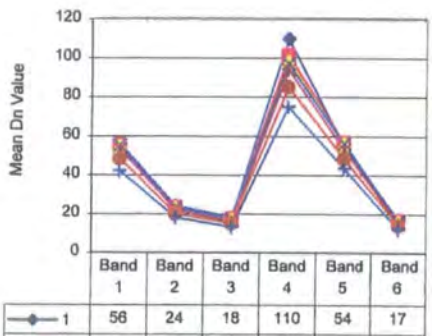
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	53	21	15	60	28	10
3	53	21	15	58	29	8
5	51	20	14	57	28	8
7	57	14	10	41	20	6

Plot 75



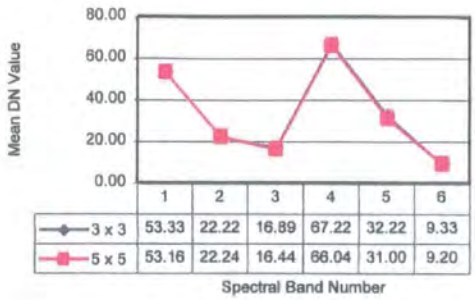
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	55	22	19	94	57	18
3	55	22	17	93	56	17
5	55	22	18	93	58	17

Plot 76



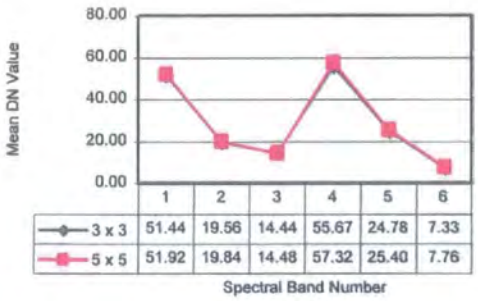
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
1	56	24	18	110	54	17
3	55	23	17	101	56	16
5	55	23	17	98	56	16
7	55	23	17	97	56	16
9	53	22	16	94	53	15
11	48	20	15	85	48	14
13	42	18	13	75	43	12

Plot 24



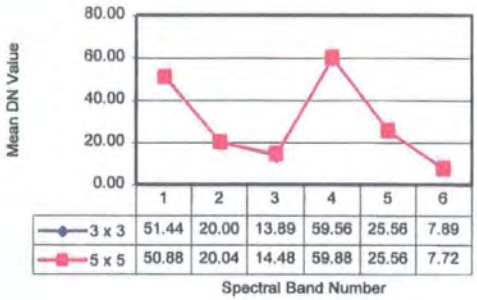
	1	2	3	4	5	6
3 x 3	53.33	22.22	16.89	67.22	32.22	9.33
5 x 5	53.16	22.24	16.44	66.04	31.00	9.20

Plot 101



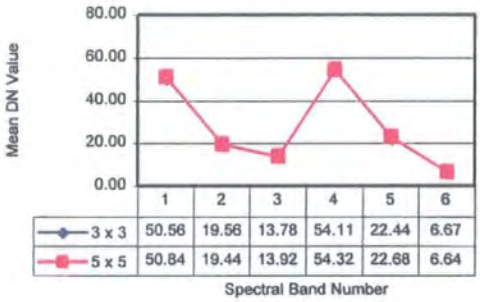
	1	2	3	4	5	6
3 x 3	51.44	19.56	14.44	55.67	24.78	7.33
5 x 5	51.92	19.84	14.48	57.32	25.40	7.76

Plot 109



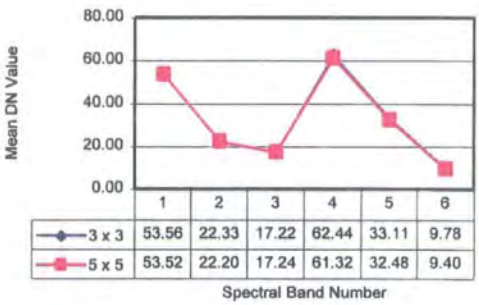
	1	2	3	4	5	6
3 x 3	51.44	20.00	13.89	59.56	25.56	7.89
5 x 5	50.88	20.04	14.48	59.88	25.56	7.72

Plot 114



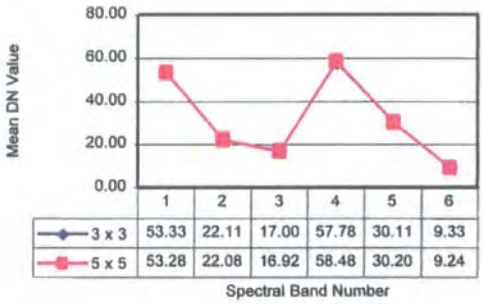
	1	2	3	4	5	6
3 x 3	50.56	19.56	13.78	54.11	22.44	6.67
5 x 5	50.84	19.44	13.92	54.32	22.68	6.64

Plot 41



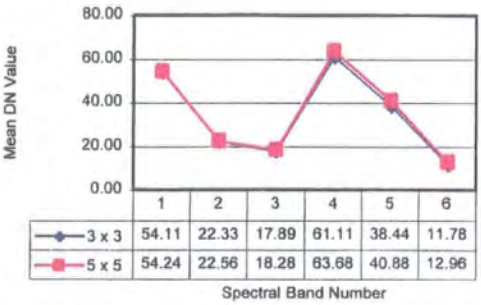
	1	2	3	4	5	6
3 x 3	53.56	22.33	17.22	62.44	33.11	9.78
5 x 5	53.52	22.20	17.24	61.32	32.48	9.40

Plot 40



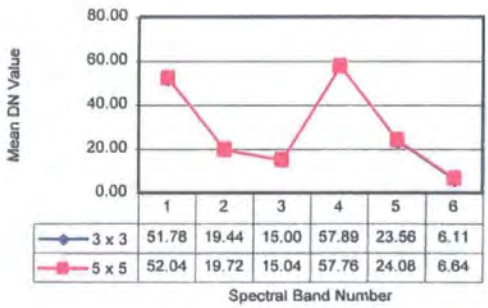
	1	2	3	4	5	6
3 x 3	53.33	22.11	17.00	57.78	30.11	9.33
5 x 5	53.28	22.08	16.92	58.48	30.20	9.24

Plot 39



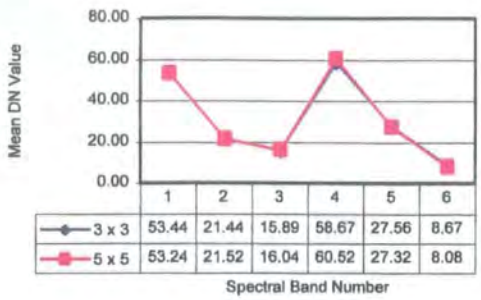
	1	2	3	4	5	6
3 x 3	54.11	22.33	17.89	61.11	38.44	11.78
5 x 5	54.24	22.56	18.28	63.68	40.88	12.96

Plot 2



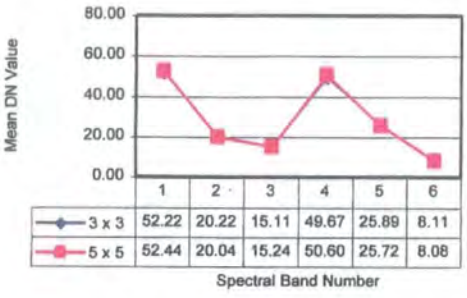
	1	2	3	4	5	6
3 x 3	51.78	19.44	15.00	57.89	23.56	6.11
5 x 5	52.04	19.72	15.04	57.76	24.08	6.64

Plot 54



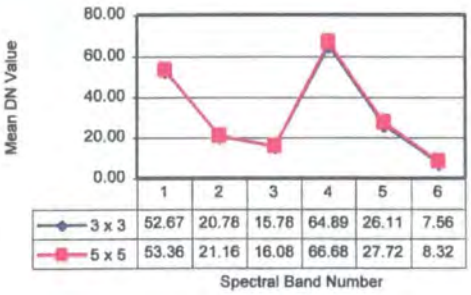
	1	2	3	4	5	6
3 x 3	53.44	21.44	15.89	58.67	27.56	8.67
5 x 5	53.24	21.52	16.04	60.52	27.32	8.08

Plot 49



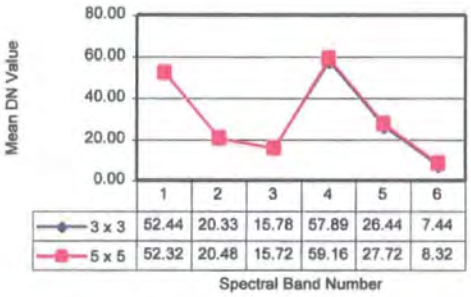
	1	2	3	4	5	6
3 x 3	52.22	20.22	15.11	49.67	25.89	8.11
5 x 5	52.44	20.04	15.24	50.60	25.72	8.08

Plot 72



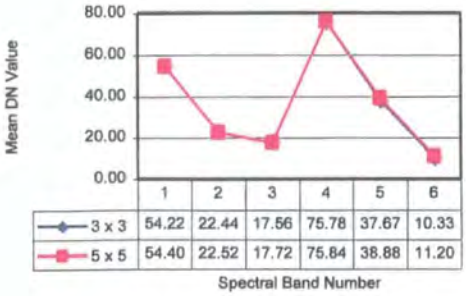
	1	2	3	4	5	6
3 x 3	52.67	20.78	15.78	64.89	26.11	7.56
5 x 5	53.36	21.16	16.08	66.68	27.72	8.32

Plot 59



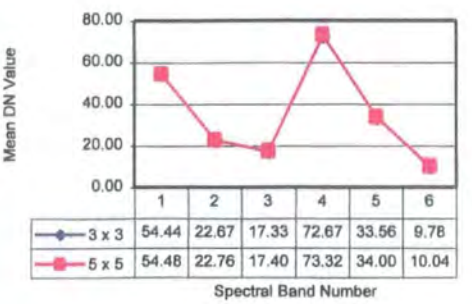
	1	2	3	4	5	6
3 x 3	52.44	20.33	15.78	57.89	26.44	7.44
5 x 5	52.32	20.48	15.72	59.16	27.72	8.32

Plot 75



	1	2	3	4	5	6
3 x 3	54.22	22.44	17.56	75.78	37.67	10.33
5 x 5	54.40	22.52	17.72	75.84	38.88	11.20

Plot 76



	1	2	3	4	5	6
3 x 3	54.44	22.67	17.33	72.67	33.56	9.78
5 x 5	54.48	22.76	17.40	73.32	34.00	10.04