UNIVERSITY OF ADELAIDE LIBRARY
17.2.94

Multi-Spectral Texture: Improving Classification of Multi-Spectral Images by the Integration of Spatial Information.

Paul J. Whitbread, B.Sc., B.E. (Hons.)

Department of Electrical and Electronic Engineering,
The University of Adelaide,
Adelaide, South Australia.

October 1992.

Awarded 1993

Abstract

Standard algorithms for segmenting multi-spectral images use only the spectral information recorded at a given pixel to classify it. In contrast, standard algorithms for segmenting monochromatic images make use of the local microstructure, or texture, of the image in a neighbourhood of each pixel for classification. This thesis aimed to fuse these two approaches by developing classification algorithms that make full use of both spectral and spatial structure and any cross correlation between them. The algorithms are based on the examination of the local micro-structure of coloured images, i.e. on the study of multi-spectral texture. The research undertaken demonstrates that algorithms making use of multi-spectral texture can be constructed that produce better classifications than standard algorithms at comparable computational cost.

In particular, the thesis presents two new families of classification algorithms for pixel classification based on multi-spectral texture. Both families outperform a conventional equal-priors maximum likelihood classifier whose inputs are only the spectral values at a pixel. The first family is based on the comparison of neighbourhood inter-band spectral covariance matrices with representative matrices for each class. Good results were obtained but a number of *ad hoc* choices and modifications were needed to produce a really efficient algorithm. The second family is based on the use of multi-layer perceptrons (a particular type of artificial neural network). Networks were constructed for a range of inputs, from spectral information available at a single pixel up to all information available in a 3x3 neighbourhood of a pixel. In all cases the networks were competitive with the standard classifier. Moreover classification accuracy improved considerably upon the addition of neighbourhood information, with best results being obtained when spatial structure was used to organize the presentation of texture inputs. Various other forms of pre- and post- processing were also shown to improve the performance of network-based classifiers. A network-based classifier that uses spatial context was proposed and shown to be viable using synthetic data.

Table of Contents

List of Figures

List of Tables

Acknowledgement

I thank my academic supervisor, Prof. R.E. Bogner for his guidance in the execution of the work for this thesis and for many intellectually stimulating discussions. His power to understand intuitively and to correlate relevant information from diverse sources would be the envy of any thinking artificial neural network.

I thank the members of the Neural Group at the University of Adelaide for sharing their understanding of the rapidly progressing field of artificial neural networks through participation in the NN Journal Club.

The Australian Research Council contributed to this work by funding equipment for image processing in EEE Dept. in the University of Adelaide, but equally importantly, by providing funds to maintain and expand the use of academic electronic mail by setting up AARNet. It is no exaggeration to say that the availability of electronic links to international researchers gave me access to preprints of papers years before publications would be available in Australia.

Financial support from the Defence Science and Technology Organisation is gratefully acknowledged, and I thank Dr D. Nichol, Dr. J. Aisbett and Dr. G. Gibbon of DSTO for encouraging and assisting me to seek that support. Dr. J. Aisbett also provided continuing encouragement along the way and ensured provision of computing equipment for much of the latter half of the work. The encouragement of Dr. G. Newsam and the members of Image Information Group within DSTO was essential for the completion of this work.

In attempting a long term project, moral support is also important. Recent PhD.'s Dall, Rockliff, Pope and their wives/girlfriends provided moral support at regular Friday evening dinners and also provided evidence that there is life after a PhD.

Finally, I thank my family for their assistance to keep going and in particular, I thank my wife, Lel, to whom this thesis is dedicated. Her encouragement and devoted, loving support made this work possible.

Declaration.

This thesis has been submitted to the Faculty of Engineering at the University of Adelaide for examination in respect of the Doctor of Philosophy.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any University, and to the best of the author's knowledge and belief contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

The author hereby consents to this thesis being made available for photocopying and for loans as the University deems fitting, should the thesis be accepted for the award of the Degree.

Paul Whitbread

Friday, 16 October 1992.

Publications

D.G. Nichol, M. Fiebig, R.J. Whatmough and P.J. Whitbread, "Some Image Processing Aspects of a Military Geographic Information System," *Australian Computer Journal*, Vol. 19, No. 3, Aug. 1987, pp. 154-160.

P.J. Whitbread and R.E. Bogner, "The Use of Pattern Recognition by Observation Correlation in Image Processing," *Proc. ISSPA87*, Brisbane, Australia, Aug. 1987, pp. 633-638.

P.J. Whitbread and R.E. Bogner, "Classification of Multi-Spectral Imagery using Neural Nets," *Proc. ASSPA89,* Adelaide, Australia, April 1989, pp. 249-253.

P.J. Whitbread and R.E. Bogner, "Neural Networks for the Classification of Multi-Spectral Texture," *Proc. ICIP89*, Singapore, Sept. 1989, pp. 713-717.

P.J. Whitbread and R.E. Bogner, "Application of neural networks to the Recognition of Texture in Multi-Spectral Images," *Proc. IPINT89*, Canberra, Australia, Dec. 1989, pp. 15-18.

He M.Y., P.J. Whitbread and R.E. Bogner, "Classification of Multi-spectral Images with Neural Network Using Binary Data," presented at *ACNN90*, Sydney, Australia, Jan. 1990.

P.J. Whitbread and R.E. Bogner, "How Can a Multi-layer Perceptron be Initialized to do Maximum Likelihood Classification?" presented at *ACNN90*, Sydney, Australia, Jan. 1990.

P.J. Whitbread and R.E. Bogner, "Using a Multi-layer Perceptron for Image Processing –Application to Classification of Multi-Spectral Images," *Proc. ISSPA90*, Gold Coast, Australia, Aug. 1990, pp. 519-522.

P.J. Whitbread and R.E. Bogner, "Internal Representation in an Image-Classifying Neural Network," *Proc. ACNN91*, Sydney, Australia, Feb. 1991, pp.254-257.

P.J. Whitbread, "Improving an Image-Classifying MLP by Teaching It Spatial Context.," presented at. *ACNN92*, Canberra, Australia, Feb. 1992.

UNIVERSITY OF ADELAIDE LIBRARY

Chapter 1

Introduction

Summary: *This chapter provides an overview of the work described by this thesis and an introduction to certain aspects of the classification of multi-spectral images. It reviews relevant work on the improvement of classification accuracy by the use of spatial information through the use of texture and the use of context. The final section highlights the sections of work that are original contributions.*

1.1 Overview

This thesis is concerned with the integration of spectral and spatial features for the classification of multi-spectral images. Some novel techniques, involving spectral covariance comparison and artificial neural networks, have been used to improve classification accuracy of spectral data with an underlying spatial structure.

The kind of image data of interest is that in which each sampling site (pixel) on a lattice has associated with it a vector of measured values. Some examples of sources of this kind of data are the Landsat 7-band thematic mapper (TM) [Engel83], the 11-band Dædalus airborne scanner, the proposed ≈200-band HIRIS (hyper-spectral) scanner for the Earth Observation System (EOS) [Dozier88] and the proposed multi-band, multi-polarization SIR/C radar [Jordon91]. The techniques covered in this research are also of interest in an area of data fusion, where each vector component could represent a measurement from a different kind of sensor. Hence in this discussion the term "multi-spectral" is meant in the most general sense.

The term "spatial feature" is used here to indicate a derived feature that is a characteristic of the local image structure. Thus measurements of spatial features are functions of the data in the neighbourhood of the pixel to be classified. For example, in classification of TM multispectral images, the spatial features, interpreted as texture, convey information to an image interpreter that is not available to a purely spectral classification system. In fusion of data from visible and radar images, the consideration of spatial features is important because of the radar image speckle phenomenon which requires some spatial (neighbourhood) processing.

Image classification is defined to mean the assignment of class labels to sub-areas of an image, in particular to pixels. The dual of image classification is image segmentation in which the boundaries defining areas containing pixels of the same class are identified. Images collected by satellite are routinely classified to identify vegetative ground-cover and surface geology. Often classification is done by measuring class-typical parameters from a reference set of pixels for each class of interest, and then using Bayesian classification involving a maximum likelihood (ML) classifier to sort pixels into their most likely classes. Classification accuracy can be measured by dividing known samples (reference sets) into two arbitrary sets: training sets and test sets. From training sets class-typical parameters are estimated The test sets are then classified using the estimated parameters and used as the basis of the accuracy measure.[1]

The combination of spectral features and spatial features for classification has been discussed in the remote sensing literature but mostly from the point of view that spatial features are monochromatic. Spatial features are usually generated from one band (only) of a multi-band image, making them essentially measures of monochromatic texture. Alternatively, some researchers have used spatial characteristics to refine a spectral classification through the use of prior information about the spatial structure of the (true) pixel labels. This latter approach is complicated by the need to estimate the priors.

There is a need to derive spatial features from a multi-spectral image as a whole, to fully use the spatial structure information in each band. For example, in a 256-band image a single band may not contain adequate spatial cues to form useful spatial features. There is a useful analogy with the extraction of objects from a sequence of images. While an object may be apparent from the images played in sequence, in many cases no individual frame contains enough data to identify the object. Similarly, while all spectral bands may contain the sought spatial information, no single band contains it all.

Two approaches have been explored in the thesis. In the first approach localized spectral (i.e. inter-band) covariance matrices were used as a feature to be classified and a measure developed to compare covariances. In the second approach artificial neural networks (ANNs) were used to learn firstly spectral characteristics and later spatial features from a neighbourhood. The majority of research effort centred on the latter class of techniques.

The techniques have been tested on a limited sample of remotely sensed images. The limitation is simply caused by the ever-present problem of obtaining detailed ground-truthed data for wide-area classification.

Matching of local inter-band covariance matrices is a promising way of classifying multi-dimensional image data, but required some *ad hoc* choices and modifications to produce an algorithm. Best results were obtained when used in a hybrid system in conjunction with a standard spectral only ML classifier that uses spectral information only.

---

[1] This is chosen as a fair way to measure accuracy, in the sense that it gives an estimate with a mildly pessimistic bias. The problems of measuring classification accuracy are discussed in the statistical literature. (e.g. [Devijver82]).

The use of a multi-layer perceptron (MLP), a particular type of ANN, was validated for classification of spectral data and later extended to use some spatial information. Classification accuracy of a (spectral only) MLP classifier equalled or exceeded the accuracy of the benchmark ML system in all cases tested. Classification accuracy improved considerably with the addition of information from a 3x3 neighbourhood of the pixel to be classified.

Various forms of pre-processing have been tested as a front-end to the MLP classifier. The most useful data manipulation involved feeding the MLP with spatial subsets (cliques) of the neighbourhood of the pixel to be classified. Whilst this data-structuring produced only a marginal improvement in accuracy, it substantially reduced the MLPs complexity.

Learning systems, such as the MLP are said to have *emergent properties* meaning they are able to capture a hidden indivisible model of the learnt information. A characteristic of learning systems with emergent properties is their inscrutability[1]. Some illumination of the internal representation in MLPs, trained for multi-spectral classification, was provided by visualization techniques. Some novel methods were explored.

Conventional classification schemes have been extended to include spatial context as a source of further information to improve classification accuracy. This thesis proposes a suitably modified MLP topology that could learn context.

In the remainder of the introductory chapter, section 1.2 describes the classification process as applied to multi-dimensional data, section 1.3 describes techniques used to identify spatial features in images while the final section, section 1.4, sets out the original contributions of this thesis.

1.2 Classification of multi-spectral images

This section briefly summarises common approaches to classification of remotely sensed multi-spectral images and establishes notation to be used in describing them. Skeletons of some standard derivations are given in order to analyze assumptions inherent in the classification schemes.

To assist formal description of supervised classification we introduce a concise definition of a digital image X

$$X = \{x_{kl} \mid (k,l) \in L,\ x_{kl} \in G\} \tag{1.1}$$

where L is a discrete finite rectangular lattice $L = \{(k,l) \mid k=1,\ldots N_k,\ l=1,\ldots N_l\}$, and G is the set of allowed discrete values for x_{kl}. We speak of each site (k,l) as a pixel and we consider the digital image to be a set of samples x_{kl} taken at pixels (k,l) of a real image which is a continuous valued function of position. It is common for a pixel to be considered to be a small rectangular area, and where this is appropriate, we consider a pixel to be centred on the point (k,l) and of a size such that all pixels are the same size and completely tile the image.

---

[1] The term inscrutable was coined in this context in a paper on the Bayesian underpinning of the maximum entropy method [Tribus88].

We define a monochromatic digital image to be an image X where each x_{kl} is a scalar (x_{kl}) restricted to take only a finite set of values, $x_{kl} \in G = \{0,\ldots,N_g\text{-}1\}$ where N_g is the number of grey levels. In general, the value of the scalar x_{kl} represents the radiance value sampled for pixel (k,l), but it may represent any measurement sampled at pixel (k,l).

We define a multi-spectral digital image to be an image X where the value of $\boldsymbol{x}_{kl}$ is a (pattern) vector with scalar components which are restricted such that $x_{kli} \in \{0,\ldots N_g\text{-}1\}$, $i = 1 \ldots M$. In general, an element x_{kli} represents the value of radiance[1] in the ith spectral band., in which case, M is determined by the sensing device: common examples are $M = 3$ for a colour (trichromatic) TV picture, $M = 4$ for Landsat MSS, $M = 7$ for Landsat TM and $M = 11$ for the Dædalus airborne scanner. However, there is no inherent restriction on the source of data represented by a component of the pattern vector. The pattern vector $\boldsymbol{x}_{kl}$ will also be referred to as a feature vector indicating that it may comprise components *derived from* measurements as well as the measurements themselves.

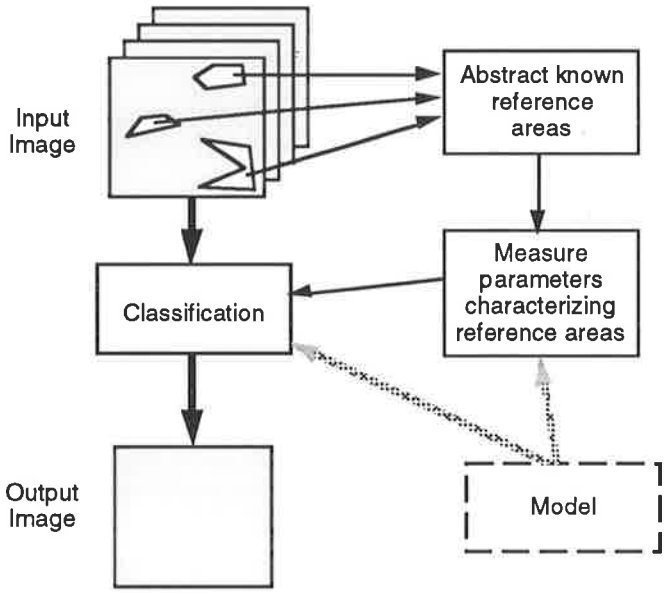


Figure 1.1: Supervised classification.

This thesis addresses the task of *supervised* classification of multi-spectral images. Figure 1.1 depicts this task in a block diagram. The aim is to classify every pixel in the given image into one of a number of classes. The input to the system is an image comprising data from a number of spectral bands (indicated by the multi-layer block) and the limited knowledge of a (human) supervisor. The supervisor identifies a number of known areas to be used as templates for classifying the whole image. The sets of reference data (training sets) taken from the known areas are used to generate parameters which characterize the classes. These

[1] Data from remote sensing platforms are radiance values which are measurements of either the brightness of reflected sun illumination or of re-radiation from the area on the ground corresponding to a pixel. For multi-spectral data, elements of the vector are observations of the same ground area made through filters, which provide spectral radiance values for defined spectral pass bands. (An actual spectral pass band is determined by the product of the filter's spectral response and that of the detector.) In analysis of image data each band defines a corresponding dimension of the data vector at each pixel.

parameters are then used to classify the whole image. Implicit in the system is a model of the data which allows the essential features of the training sets to be distilled into a few parameters.

Unsupervised classification, which amounts to clustering of data, is also used in processing remotely sensed images. However, it is not considered here as a subsidiary goal of this thesis was to look for class characteristics that could be useful on more than just the original image used for training. Clustering systems produce results which are a function of the data that appears in the particular image being processed. Such data dependency, while flexible, runs counter to the concept of being able to transfer class descriptions from one image to another in the form of (parameterized) spectral/spatial signatures.

We define classification of an M-band multi-spectral image of size $N = N_k \times N_l$, to be a mapping

$$f: (\mathbb{R}^N)^M \rightarrow \Omega^N \tag{1.2}$$

where IR is the set of real numbers and $\Omega = \{\omega_r, | r = 1,\ldots R\}$ is the set of all possible classes. A "classified image" is an image consisting of class labels $Z = \{z_{kl} | (k,l) \in L, z_{kl} \in \Omega\}$ where $z_{kl} = \omega_r \Rightarrow x_{kl}$ comes from class ω_r. We assume that the lattice L has associated with it a true but unknown set of labels $T = \{t_{kl} | (k,l) \in L, t_{kl} \in \Omega\}$. During classification we want to assign labels Z to L, that are in some sense a best guess of T, given that we are only able to observe image data X. The best assignment, in the Bayesian sense, is the particular Z which minimizes the cost of misclassification:

$$\min_{Z \in \Omega^N} \left[\sum_{T \in \Omega^N} \{\alpha(Z,T)\, P(T|X)\} \right] \tag{1.3}$$

where $\alpha(Z,T)$ is the loss associated with the assignment of labels Z when the true labels are T, and $P(T|X)$ is the probability that the true labels could be generated, given the data. This is almost always rewritten using Bayes rule as:

$$\min_{Z \in \Omega^N} \left[\sum_{T \in \Omega^N} \{\alpha(Z,T)\, P(X|T)\, P(T)\} \right] \tag{1.4}$$

where the denominator term of $P(X)$ has been neglected because it does not change over the range of parameters.

In conventional schemes of multi-spectral classification, (1.4) is made computationally feasible by making two assumptions and choosing an appropriate loss function:–

ASSUMPTION A1: The description process is local; i.e. no characteristics from pixel (k,l) are affected by characteristics of pixel (i,j) $i \neq k$, $j \neq l$.

$$P(X|T) = \prod_{(k,l) \in L} P(x_{kl}|T) \tag{1.5}$$

ASSUMPTION A2: The pixel's labels are independent; i.e. there are no inter-pixel dependencies and

$$P(T) = \prod_{(k,l) \in L} P(t_{kl}) \tag{1.6}$$

When it is desirable to incorporate spatial properties these assumptions will not be appropriate (see section 1.3).

LOSS FUNCTION L1: In situations where the classes have no arithmetic relationship to each other (e.g. land, sea, trees), and hence measures such as Euclidean distance are meaningless, a useful loss function is to assign zero loss for correct joint assignment and unit loss for incorrect assignment.

$$\alpha(Z,T) = \left\{ \begin{array}{l} 0 \text{ if } z_{kl} = t_{kl} \ \forall (k,l) \in L \\ 1 \text{ otherwise} \end{array} \right\} \tag{1.7}$$

Given assumptions A1 and A2, (1.4) becomes

$$\min_{Z \in \Omega^N} \left[\sum_{T \in \Omega^N} \left\{ \alpha(Z,T) \prod_{(k,l) \in L} P(x_{kl}|t_{kl}) P(t_{kl}) \right\} \right] \tag{1.8}$$

Using the loss function L1, the Bayes decision rule is to choose, pixel-by-pixel, the z_{kl} that satisfies

$$\max_{z_{kl} \in \Omega} [P(x_{kl}|z_{kl}) P(z_{kl})] \tag{1.9}$$

(see [Haralick83] for a proof). If we define the particular set of discriminant functions, $g_r(x) = P(x|\omega_r) P(\omega_r)$, $r = 1,\ldots R$, is possible to rewrite (1.9) as

$$z_{kl} = \omega_r \Leftrightarrow g_r(x_{kl}) > g_s(x_{kl}) \ \forall \ r,s = 1\ldots R, \ r \neq s \tag{1.10}$$

which leads to a simple scheme for implementation (figure 1.2). The concept of discriminant functions is also of interest in visualizing how M-dimensional spectral space is partitioned into regions corresponding to classes. This will be relevant when discussing the functioning of image-classifying artificial neural networks.

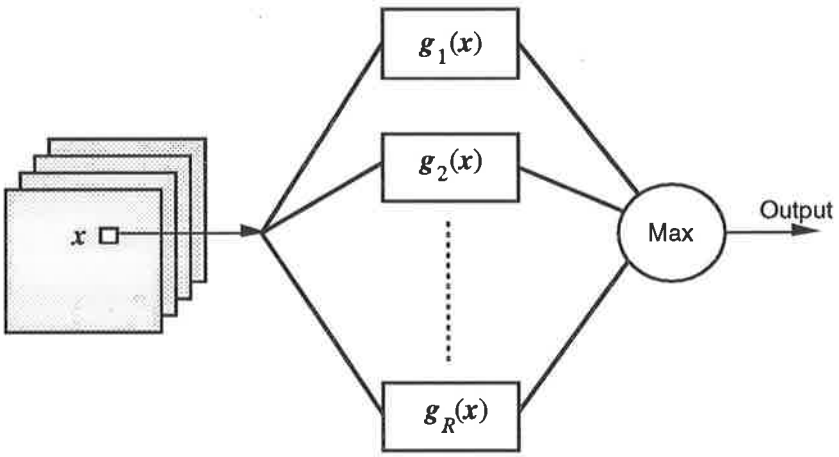


Figure 1.2: The classification process; pixel-by-pixel class assignment.

Under conventional Bayesian maximum likelihood classification, the discriminant function is calculated by modelling the probability density functions of classes as multi-variate normal with known parameters. The parameters, $\boldsymbol{m}_r$ (the M-vector of means) and $\boldsymbol{C}_r$ (the $M \times M$ covariance matrix) are estimated by observing the pattern vectors of elements of a training set. A further common assumption is that of uniform prior probabilities; i.e. equally likely classes

are assumed (especially when there is no information about a particular class). These assumptions lead to a discriminant function g_r of the form

$$g_r(x) = -\log|C_r| - (x - m_r)^t C_r^{-1}(x - m_r) \quad (1.11)$$

(see [Richards86] for derivations).

The assumptions of multi-variate normal and uniform priors are clearly not always correct but in practice lead to surprisingly robust classifiers. However, there is a specific case when the multi-variate normal assumption leads to a poor classifier: when the probability distribution function of a class is multi-modal. In this case the model and its parameters, the mean and covariance, do not provide a good description of the class characteristics. The most common cause of this is a heterogeneous textured class.

Where the multi-variate normal distribution assumption is clearly violated, one approach is to decompose classes into sub-classes which are more nearly normal [Richards86]. For classes with multi-modal distributions, pre-classification clustering of training set data can be used to guide manual splitting of classes, which are re-combined after classification. Subclasses based purely on spectral characteristics are often distinguished from the classes involving human interpretation by using the terms *spectral classes* and *information classes*. In general, manual intervention to split classes containing highly dimensional (many band) data with complex distributions is not convenient. Automatic clustering procedures incorporate a number of "tuning" parameters which require prior knowledge of the data. Images with many bands can present problems in choice of clustering parameters. The classification scheme explained in Chapter 3 using an MLP does not rely on the nature of the class probability density functions and hence does not require the splitting of classes.

While ML is the most accurate classifier in most cases, it is not favoured when the dimensionality of the pattern vectors to be classified is large. This is because when ML is implemented on a general purpose serial computer the computing requirement increases as $O(M^2)$, where M is the dimensionality of the pattern vector (equal to number of spectral bands in the simplest cases). Special purpose pipeline processors (e.g. [Swain80], [Ramapriyan85]) have been constructed for multi-spectral classification, but because of their expense and limited applicability to other problems, are not widely available. It is not uncommon to throw away bands to achieve usable speeds of classification and selection of an N-band subset from M bands has been a subject of research [Sheffield85]. Techniques such as band ratioing and linear transforms [Swain78] are also used to try to concentrate information in a lesser number of dimensions.

1.3 Using spatial features to classify multi-spectral images

The main motivation for trying to combine spectral and spatial information is that a human photo-interpreter is able to use spatial cues to improve spectral classification accuracy. With reference to classification of urban areas, Jensen remarked that "Manual photo interpretation of such heterogeneous surfaces is generally on a synergistic evaluation of (a) context, (b) edges, (c)texture and (d) tonal variation" [Jensen79a]. In relation to interpretation of (simulated) TM

images, Wharton wrote "The consistency of visual interpretation was attributed in part to the ability of the analyst to integrate knowledge of color, contrast, and tonal properties of each category and to consider the local distribution of spectral categories that serves to confirm or contradict a given pixel classification" [Wharton87].

Spectral/spatial integration of data from remote sensing platforms has been of interest since the inception of such data collection. The general practice of processing spectral information alone to give classified images has resulted from the perceived, and in most cases real, requirement for very high computing power to process spatial data. Renewed interest in spectral/spatial integration comes about because of the widespread availability of powerful computer workstations, and also because of the possibility of using the new computational paradigm of the artificial neural network to significantly reduce the effective computing time.

Investigation of spectral-spatial integration is also warranted by the advances in spatial resolution of multi-spectral satellite sensors. With better spatial resolution, it will be more common for particular classes to comprise heterogeneous (i.e. textured) data with considerable (possibly systematic) variation, since the optics will no longer average out unwanted fine detail [Woodcock87], [Cushine87]. What detail is unwanted is a function of the problem domain. In this thesis the problem domain is taken to be that of wide area classification. For example, the class "urban" could be broken down into red roofs, green lawns, and black pavement, in one problem domain but in another, coarser, problem domain "urban" is a single textured class. Another example is provided by crops grown in a particular geometric arrangement, giving the possibility alternating pixels but never-the-less constituting a homogeneous class.

Construction of a classification scheme to include spatial information from multi-spectral images can take two main forms which amount to pre-processing and post-processing. Pre-processing can be used to produce some extra features that embody information about the neighbourhood of the pixel to be classified[1]. This process measures the structure of expected within-class local variation of pixels' values which effectively constitutes "texture". Post-processing can be used to refine the (classified) output image using prior information about its expected structure. For example, prior information could indicate that class A often occurs surrounded by class B, but not vice versa. This kind of knowledge measures a spatial "context". Context measurement can form the basis of an iterative scheme in which the accuracy of classification is refined with each successive estimate.

The true division between texture-based and context-based schemes is not so well defined. In terms of the Bayesian image classification described in (1.4), allowing classes to have texture can amount to the relaxation of assumption A1, and assumption A2, which disallows spatial relationships between true labels in T, must be abandoned if we are to acknowledge the usefulness of context. The blurring of distinction between texture and context occurs when we consider the definition of texture as "within class variation" where we may be using information rather than spectral classes. It is possible that within an *information* class variation

[1] Pre-processing can also be used to reduce the number of features when handling them would represent too great a computational load – getting rid of one band is a crude form of pre-processing.

appears as multiple adjacent *spectral* classes. In this situation, texture at the *information* class level appears as context at the *spectral* class level. Of course, in a situation where the classification system does not require splitting of information classes, the difference is apparent.

In this introductory chapter, processes in which features are derived purely from the input image structure will be considered under the heading of "texture" and processes in which there must be an estimate of the classified image before features can be derived will be considered under the heading of "context".

1.3.1 Texture

The Webster's dictionary definition of texture is "the visual or tactile surface characteristics and appearance of something". Rao points out [Rao90], texture has a rich range of meanings in the divers fields of Engineering, Natural Sciences, Art and Design. Here, only visual aspects are of interest.

Horn defines visual texture as "detailed structure in an image that is too fine to be resolved, yet coarse enough to produce fluctuation in the grey levels of neighboring cells" [Horn86]. In this work texture is defined as "structure in the image which is a characteristic of a class, but of a scale too small to be of interest in the labelled (classified) image domain." If we acknowledge that this structure exists then assumption A1 is contradicted. To create a workable classification system, we assume that by augmenting the original pattern vector with enough spatially dependent measurements from a pixel's neighbourhood, A1 becomes approximately true.

The most common approach to combining texture information with multi-spectral information is to use monochromatic texture of one of the spectral bands of an image to augment the spectral information. This involves identifying measures of monochromatic texture which are used as extra features during classification.

There have been many attempts to characterize monochromatic texture both stochastically and syntactically; the computer vision literature on (monochromatic) texture is vast and the comments here are intended only as a brief survey of those methods that have potential application in remote sensing[1]. Methods range from the use of power spectral[2] density functions, the use of spatial grey-level dependence or difference matrices, to the use of filters to detect texture primitives or the boundaries between texture primitives. All approaches try to reduce the large amount of data about a pixel's neighbourhood to a feature vector of manageable dimensions.

To extract the texture information for a particular pixel, what is required is a convenient compact form of the joint spatial/spatial frequency information relating to each pixel. Some methods try to do this quite explicitly. In the literature, there are many attempts to use Fourier transform methods and autocorrelation to produce such a representation (e.g. see [Conners80]),

---

[1] For a more general survey, see, for example vanGool *et al.* [vanGool85] and the yearly Rosenfeld bibliographies(e.g. [Rosenfeld90]).

[2] "Spectral" here refers to the spectrum of the spatial frequencies.

but these are hardly compact. The Wigner transform has the capacity to capture spatial/spatial frequency characteristics in a compact form [Reed90] but is computationally expensive and is subject to some problematic artifacts. Pentland and others have shown that it is viable to use fractals to provide a local characterization of texture [Pentland84], [Peleg84], [Lundahl86].

Most computationally feasible approaches to texture are based on first or second order statistics and were inspired by the early perceptual work of Julesz [Julesz62], [Julesz65], [Julesz73]. At that time, on the basis of experiments with human subjects, Julesz conjectured that textures with the same first and second order statistics would be perceived as being identical. (Later, counter-examples were found.) Consistent with this conjecture, Haralick *et al.* [Haralick73] defined the "spatial grey level dependence" or "co-occurrence" matrix and derived features from it that could be used to discriminate textures. (For reviews of other matrix measures see [Weszka76], [Conners80]).

To define the co-occurrence matrix, consider two pixels in an image, x_1 and x_2, separated by the vector (d_1,d_2), such that $x_1 = x_{k,l}$ and $x_2 = x_{k+d1,l+d2}$. The Julesz conjecture implies that if we consider the textured image to be a realization of random variables, drawn from a 2-dimensional stationary, ergodic process then it is completely characterized by the discrete joint probability function $P(x_1, x_2)$ of the two random variables of x_1 and x_2. Thus the probability of observing the grey levels i and j at fixed displacement (d_1,d_2) is independent of l and k, and is given by

$$P(i,j) = P(i,j; d_1,d_2) = \text{Prob}\ \{x_{k,l} = i,\ x_{k+d_1,l+d_2} = j\} \tag{1.12}$$

This probability function can be approximated by the normalized co-occurrence matrix

$$\widehat{P}(i,j) = \frac{c(i,j)}{N} \approx P(i,j) \tag{1.13}$$

where the co-occurrence matrix over the neighbourhood η is defined by

$$c(i,j) = c(i,j; d_1,d_2) = \#\{(k,l) \in \eta,\ x_{k,l}=i,\ x_{k+d_1,l+d_2}=j\} \tag{1.14}$$

the operator # means count or cardinality and N, the total number of counts is given by

$$N = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} c(i,j) = \#\{\eta\} \tag{1.15}$$

Table 1.1 gives definitions required to calculate Haralick's texture features along with the simplifications of Unser which were intended to reduce the computational requirement [Unser86]. Even with simplification, calculation of Haralick's texture measures on a pixel by pixel basis can be a formidable computational overhead.

Table 1.1a: Co-occurrence matrix and sum-and-difference histogram definitions for a fixed displacement (d_1,d_2) or equivalently (d,ϑ)

Co-occurrence Matrix form	Sum and Difference Histogram form
$c(i,j;d_1,d_2) = c(i,j)$	$h_s(i;d_1,d_2) = h_s(i);\ h_d(j;d_1,d_2) = h_d(j)$
$c(i,j) = \#\{(k,l) \in \eta,\ x_{k,l}=i,\ x_{k+d1,\,l+d2}=j\}$	$h_s(i) = \#\{(k,l) \in \eta,\ x_{k,l} + x_{k+d1,\,l+d2} = i\}$
	$h_d(j) = \#\{(k,l) \in \eta,\ x_{k,l} - x_{k+d1,\,l+d2} = j\}$
$P(i,j) \approx \widehat{P}(i,j) = \frac{c(i,j)}{N}$	$\widehat{P}_d(j) = \frac{h_d(j)}{N};\ \widehat{P}_s(i) = \frac{h_s(i)}{N}$
where $N = \#\{\eta\} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} c(i,j)$	$N = \#\{\eta\} = \sum_{i=1}^{N_g} h_s(i) = \sum_{j=-N_g+1}^{N_g-1} h_d(j)$

Table 1.1b: Texture features for a fixed displacement (d_1,d_2) or equivalently (d,ϑ)

Texture Feature	Co-occurrence Matrix form	Sum and Difference Histogram form
(Sum) Mean	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} i\,\widehat{P}(i,j)$	$\frac{1}{2}\sum_{i=2}^{2N_g} i\,\widehat{P}_s(i)$
(Sum) Variance	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} (i-\mu)^2\,\widehat{P}(i,j)$	$\frac{1}{2}\left(\sum_{i=2}^{2N_g} (i-2\mu)^2\,\widehat{P}_s(i) + \sum_{j=-N_g+1}^{N_g-1} j^2\,\widehat{P}_d(j)\right)$
Angular second moment (energy)	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} \left(\widehat{P}(i,j)\right)^2$	$\sum_{i=2}^{2N_g} \left(\widehat{P}_s(i)\right)^2 + \sum_{j=-N_g+1}^{N_g-1} \left(\widehat{P}_d(j)\right)^2$
Correlation	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} (i-\mu)(j-\mu)\,\widehat{P}(i,j)$	$\frac{1}{2}\left(\sum_{i=2}^{2N_g} (i-2\mu)^2\,\widehat{P}_s(i) - \sum_{j=-N_g+1}^{N_g-1} j^2\,\widehat{P}_d(j)\right)$
Entropy	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} -\widehat{P}(i,j)\log\left(\widehat{P}(i,j)\right)$	$-\sum_{i=2}^{2N_g} \widehat{P}_s(i)\log\left(\widehat{P}_s(i)\right) - \sum_{j=-N_g+1}^{N_g-1} \widehat{P}_d(j)\log\left(\widehat{P}_d(j)\right)$
Contrast	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} (i-j)^2\,\widehat{P}(i,j)$	$\sum_{j=-N_g+1}^{N_g-1} j^2\,\widehat{P}_d(j)$
Homogeneity	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} \frac{1}{1+(i-j)^2}\,\widehat{P}(i,j)$	$\sum_{j=-N_g+1}^{N_g-1} \frac{1}{1+j^2}\,\widehat{P}_d(j)$
Cluster Shade	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} (i+j-2\mu)^3\,\widehat{P}(i,j)$	$\sum_{i=2}^{2N_g} (i-2\mu)^3\,\widehat{P}_s(i)$
Cluster prominence	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} (i+j-2\mu)^4\,\widehat{P}(i,j)$	$\sum_{i=2}^{2N_g} (i-2\mu)^4\,\widehat{P}_s(i)$

A further practical problem arises because the N_gxN_g co-occurrence matrices are functions of two displacement parameters and hence the texture features only refer to a particular set of parameters. In most applications of it is only possible to try a small number of the possible parameter values. Further, the choice of window size over which the co-occurrence matrix is estimated is rather arbitrary. Hence, texture classification routines are based to a large extent on empiricism. Despite this, Haralick's texture features are widely used to extract classifiable features from grey level images in the fields of computer vision (e.g. [Siew88], [Gotlieb90]) and remote sensing (e.g. [Weszka76], [Marceau90], [Barber91]). Their use in conjunction with multi-spectral data is discussed in a separate section below.

In related work, some researchers have used co-occurrence (or similar) matrices with different sets of texture features, or without explicitly extracting any features. Vickers and Modestino classified co-occurrence matrices using log-likelihood [Vickers82], Parkkinen *et al.* classified the matrices using subspaces [Parkkinen86] and Oja by regularity measurement through the use of the kappa statistic [Oja89].

Laws' thesis [Laws80] detailed a quite different approach in which an image was pre-processed by 3x3, 5x5 or 7x7 filter masks (convolution kernels) and then statistics collected over larger areas (e.g. 15x15) were used, after some transformation, as "energy" features for discrimination. Laws' selection of the small filter masks was essentially empirical, but in further work Ade derived an alternative set of "eigen" filters on the basis of a theoretical consideration of the 3x3 neighbourhood [Ade83], while Chen used the method of least squares to derive a set of filters matched to the particular textures that he wished to identify. Aach *et al.* restated the method in terms of quadrature filter pairs [Aach88]. Unser has developed these methods using nonlinear filters to exploit spatial localization of "energy" at different scales [Unser86b], [Unser89]. Hsiao and Sawchuk used Laws' texture energy feature set as the basis for segmenting a textured image using relaxation techniques [Hsiao90].

In a survey of some widely cited statistical texture feature sets, du Buf *et al.* [Buf90] showed that the most efficient feature sets, for their set of monochrome test images from the Brodatz collection [Brodatz66], were (in order of merit) those provided by Haralick *et al.* [Haralick73], Laws [Laws80] and Unser [Unser86b]. An earlier survey of texture algorithms specifically tested on monochromatic remotely sensed images was made by Weszka *et al.* [Weszka76], but it does not assess the more modern techniques.

A stochastic model-based approach to texture can be more powerful that a purely statistical approach, and a number of such models have been explored by various researchers. Physically based processes provided models for early examples of this approach [Schachter78]. Later models mostly fall into the class of discrete-index Markov-type random processes (also called Markov random field models). These are defined at length by Derin and Kelly who emphasise equivalence of Gibbs and Markov representations for a finite lattice [Derin89]. Markov random field (MRF) based models to analyze texture have appeared under the names of Markov models [Cross83], Gauss-Markov models [Chellappa85],[Cohen91], Gibbs random

field models [Derin86], [Derin87], [Pickard91], [Pickard91a][1], linear predictive models [Kobatake86] and autoregressive models [deSouza82], [Kashyap86], [Kartikeyan91]. Longstaff and Howard have also proposed the use of a Boltzman machine (a form of ANN) to learn the parameters of an MRF in order to characterize texture [Longstaff90]. The use of MRF models in the definition of "spatial context" is discussed in section 1.3.

Some recent approaches to texture identification have been based on the continually evolving understanding of the human visual system. This includes approaches based on evidence provided by perceptual experiments; e.g. Julesz's texton theory (incorporating the theory of selective attention) [Julesz86], [Voorhees87] and also schemes seeking to use neurophysiological descriptions, most of which can be traced back to the work of Hubel and Wiesel on the cat's visual system [Hubel59]. On the basis of observed retinal receptive field profiles, Daugmann has attempted to measure texture features using Gabor functions, a particular form of wavelet function [Daugman89], [Daugman90]. Following the same reasoning, Bovik *et al* have suggested directionally tuned narrow band filters to capture the information [Bovik90], [Bovik91]. Skrzypek suggested a specific multi-layer architecture for the discrimination of texture based on neurological examination of the mammalian visual cortex [Skrzypek87].

Some researchers have argued that a general characterization of texture requires the description patterns that repeat on a scale larger than small neighbourhoods; i.e. we need a description of the spatial relationships of micro-constructs inherent in the formation of macro-texture. Wang *et al.* used the statistics of texture primitives (connected regions satisfying certain properties), rather than of grey levels, to discriminate textures [Wang81]. Lu and Fu sought to derive grammars that described the relative location of uniform regions [Lu79], [Fu86]. Vilnrotter developed relational descriptions of "edge" micro-features [Vilnrotter86]. Rao has proposed a sophisticated texture analysis system designed to handle both macro- and micro-texture elements [Rao90]. However, the dichotomy between micro and macro structure remains unsatisfyingly imprecise and domain dependent. Even the work on textures at multiple scales (e.g. [Peleg84], [Burt83], [Unser89]) makes assumptions about the scale at which we wish to acknowledge detail. To avoid this problem this thesis deals with the restricted problem domain of remotely sensed images and textures considered here are universally thought of as microstructures comprising pixels of different tonal values or colours.

While there is a large body of literature on computer vision aspects of monochromatic texture, there is not a correspondingly large computer vision literature on colour texture or multi-spectral texture. One of the main reasons is the problem of colour constancy; i.e. the stable perception of colour over varying light conditions [Swain90]. Some current research is addressing this problem (e.g. [Hurlbert88], [Tominaga88], [Swain90], [Poggio90]). Another reason for the lack of attention is the inherent trade-off between extra colour bands and extra spatial resolution. Spatial resolution is usually considered more important in computer vision

[1]The work of Pickard *et al.* is particularly interesting since it provides a framework for unifying co-occurrence matrix descriptions and Markov random field descriptions.

tasks. To measure colour texture we would like both good spectral resolution and good spatial resolution.

Some work based on human perceptual models addressed the problem of creating colour textures [Gagalowicz81], and an efficient model for colour textures was proposed by Gagalowicz *et al.* [Gagalowicz86]. In the 1986 work, colour texture was characterized by clustering the data in colour space to L clusters which were then treated in the same way as L grey levels in a monochrome texture system. The justification for clustering was an observation that in many situations colour data does not fill the 3-dimensional colour space uniformly and that reduction of dimensionality of the data is possible with minimal loss of information[1]. It is not clear how the clustering algorithm used by Gagalowicz *et al.* ensures that proximity of two colours in a 3-dimensional colour space is preserved in the transformed 1-dimensional space. However, they were able to apply their model successfully to colour texture synthesis. Klinker *et al* [Klinker88] devised methods of colour object recognition which also relied on the limited domains of naturally occurring colour values in red-green-blue space. It is necessary to exercise care in translating processes defined with respect to human visual colour space to multi-spectral space. A number of transformations of images apparently remove no information when assessed by human observers, but may delete information that can be identified by machine. The most trivial example of this is the ability of machine sensors to use wavelengths to which the human visual system is not sensitive. However, in a broader sense it is possible that a computer vision system or a multi-spectral sensing system may contain class discriminating information, e.g. high frequency spatial information, that would be discarded in a human visual system.

In the field of remote sensing the use of monochromatic texture measured from multi-spectral images has been widely studied as a possible source of extra information to aid spectral classification. In this problem domain, the lack of colour constancy can be overcome to a large extent by calibrated radiometric corrections because sensing is from a platform with a known position with respect to a stable illuminator, the sun.

In an early paper describing the use of co-occurrence matrices, Haralick *et al.* reported combining multi-spectral data with the texture features "angular second moment" (ASM), "contrast", "correlation" and "entropy", measured from data in a single spectral band (see Table 1.1). They showed a data-dependent improvement in accuracy over purely spectral classification [Haralick73]. Further data using 32 texture features derived from the co-occurrence matrix was presented in [Haralick74], again showing significant improvement of spectral/spatial feature classification versus the spectral-only classification. Jensen experimented with improving the identification of "urban fringe" from multi-spectral Landsat data by using pattern vectors comprising three components from spectral measurements plus one texture feature derived from a single band [Jensen79a]. Out of four texture measures tested

[1]This was recognized in the context of colour coding for NTSC television standard in the early 1950's (see especially [NTSC54]). The same observation was the basis of work in digital colour image coding (e.g. [Pratt71]) and colour image restoration [Hunt84], [Galatsanos89].

for use as the augmenting component, ASM gave the best improvement in accuracy, but Jensen commented on the large cost in terms of computing requirement. It is noteworthy that, in general, this early work used simpler classification methods than ML both to reduce the computing requirement and also because the probability distribution function of various texture measures was not known and could not be guaranteed to resemble a normal distribution.

Shih and Schowengerdt [Shih83] used texture features related to the maxima/minima features of Mitchell *et al.* [Mitchell77] to help classify desert landforms where the spectral information was insufficient. Texture features were derived from band MSS-5 Landsat data only and used to augment spectral data in an ML classification scheme. Three thresholds for the maxima/minima process and the window size were selected empirically. Two of the classes they sought to identify were not separable by spectral means but could be identified when texture was included. In many ways this highlights the fact that the expectations of a user influence whether the data (spectral and/or spatial) will be sufficient to allow accurate classification. Clearly a class described by spatial characteristic alone, e.g. "hill slopes", is not a good candidate for spectral classification. Hence, in making the claim that spectral/spatial integration is important, this thesis is arguing, in part, that users now expect to be able to describe classes using a more flexible set of attributes than just spectral reflectance.

Franklin and Peddle [Franklin89] used Haralick's "entropy" feature derived from each of 4 bands in 4 directions to augment Landsat MSS with some improvements in accuracy dependent on the nature of the class. They found that for classes that are controlled structurally or topologically texture was a better class discriminator than multi-spectral data. Classes with little local variability such as "water" had marginally decreased accuracy.

In a 1979 review article, Haralick defined texture as an organized area phenomenon with two "dimensions"; the *tonal primitives*, and the *arrangements* of those primitives [Haralick79]. A logical extension of this definition replaces the concept of tonal primitives with multi-spectral primitives giving "multi-spectral texture". This term was first coined by Rosenfeld *et al.* [Rosenfeld82] in a paper describing an algorithm to extract textural information from M bands of an image using matrix methods. The 2-dimensional $N_g \times N_g$ co-occurrence matrix becomes a $2M$-dimensional matrix for an M band image, making it unmanageable; e.g with only $2^3 = 8$ grey levels and 2 bands, the resulting matrix has $(2^3)^4 = 4096$ elements. Rosenfeld *et al.* reduced this complexity by using absolute difference histograms instead of co-occurrence matrices, making the resulting matrix M-dimensional. The absolute difference histogram, A, over the neighbourhood η, is defined by

$$a(i; d_1,d_2) = \#\{(k,l) \in \eta,\ |x_{kl} - x_{k+d1\ l+d2}| = i\} \qquad (1.16)$$

In one dimension (band) the difference histogram A_δ is an N_g-vector whose ith element is the number of pairs of pixels in relative position $\delta = (d_1,d_2)$ that have an absolute grey level difference of i. For M bands, A_δ is a M-dimensional scatter plot. Rosenfeld *et al.* showed that in principle, pairs of two-band textures exist that can be discriminated easily when features are based on a two-bands A_δ but that are hard to discriminate based on single band features. In Chapter 2 this possibility is discussed and the discrimination power of multi-band statistics is shown to be useful.

Another approach to simultaneous use of multi-spectral and spatial information was taken by Eklundh *et al.* [Eklundh86] who used an associative neural network to cluster the combined data into information classes in an unsupervised classification system. Other unsupervised systems have been proposed by Grossberg based on the ART neural network [Grossberg87a]. Hepner *et al.* [Hepner90] used ANNs for supervised classification of remotely sensed data and mentioned inclusion of spatial data but gave no details.

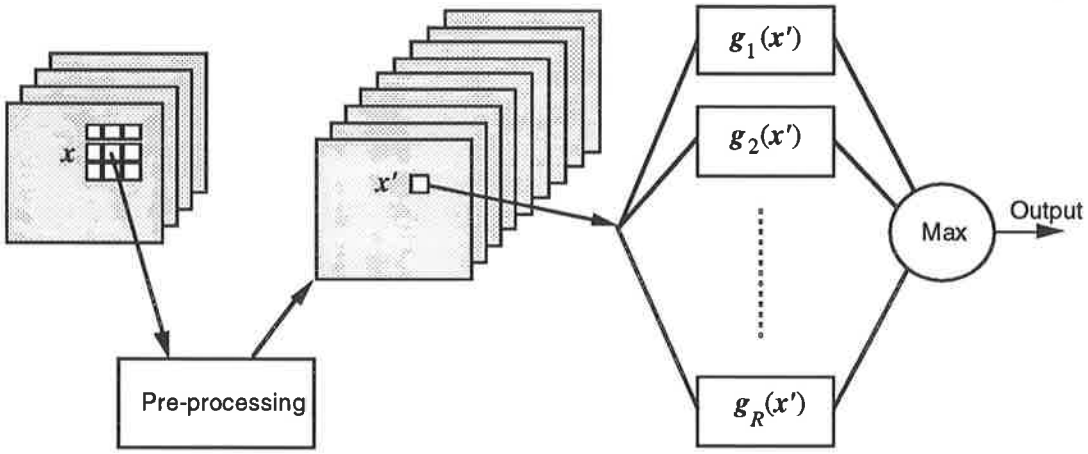


Figure 1.3: Augmented ML classification.

In each case described in this section, classification using new features derived from texture amounts to a pre-processing exercise followed by classification (figure 1.3). There are three obvious problems resulting from the pre-processing operation. Firstly, there is an increase in the number of components in the pattern vector to be classified. As remarked earlier, increasing the number of features, M, increases the amount of computation required to classify using the ML scheme by a factor of M^2. Another problem is that there is no guarantee of normality of the probability distribution function of features generated by pre-processing. It was the consideration of these two problems that lead to the exploration of the multi-layer perceptron for classification (Chapter 3). Finally a blurring of the information location occurs since the features incorporate data from neighbourhoods instead of individual pixels. This problem is difficult to address without additional information. One approach is to build a classification system that takes into account not only the raw input data from input image but also the expected form of the output image. This expectation is often called *context*.

1.3.2 Context

The dictionary definition of "context" is "what precedes or follows word or passage especially as throwing light on its meaning". Here, we are interested in the spatial surrounds or neighbourhood of a pixel. Haralick and Joo explain that "the effect of context is that a pixel's most probable interpretation when viewed in isolation changes when viewed in some context." [Haralick86]. The use of context to improve classification accuracy is inherently an activity that takes place after classification, or at least after the first estimated classification as part of an iterative scheme, since to assess how like/unlike a pixel's label is to that of its

neighbours requires at least a rough estimate of the neighbours' classes. The use of prior information about the model of a desired output image is implicit in the use of context.

We define *spatial context* for the purpose of this thesis study to be the spatial relationship of pixels of one class to the pixels of other classes. The recognition that such a relationship exists contradicts assumption A2. A convenient, workable replacement for A2 is the assumption of Markov random field [Derin89], usually with a small neighbourhood:

ASSUMPTION A2a: The pixel's labels dependent only on labels in a defined neighbourhood η:

$$P(T) = \prod_{(k,l) \in L} P\left(t_{kl} | T_{kl}^{\eta}\right) \tag{1.17}$$

where T_{kl}^{η} is the set of labels of pixels in the neighbourhood η of (k,l).

Most of the stochastic approaches to context described here use this assumption either implicitly or explicitly.

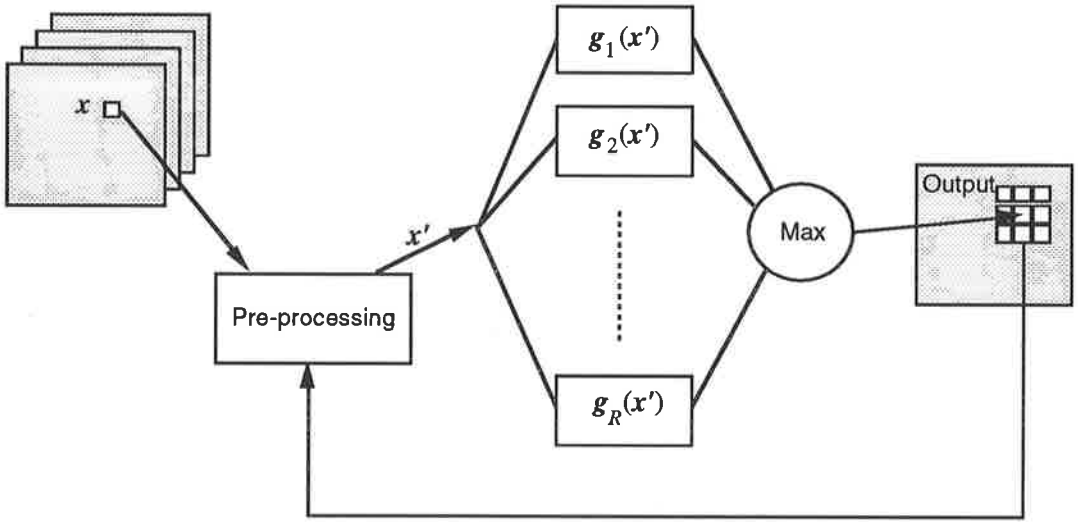


Figure 1.4: Contextual classification.

The simplest approach to using context is to filter the classified image using a scheme that is suitable for an unordered set; e.g. majority filter. Richards *et al.* discuss a sophisticated post processing scheme based on pixel relaxation that is suitable for use with image data from remote sensing [Richards81], [Richards81a]. Their modification to the probabilistic relaxation method of image labelling [Zucker78] allowed initial pixel labels generated by a spectral classification to influence the relaxation throughout the process. The modification required an empirically determined weighting factor which changed during relaxation in accordance with a suitable schedule.

Swain, Vaderman and Tilton used compound decision theory to develop a method for incorporating prior information about the probability of allowable neighbourhood configurations into the classification process [Swain81], [Tilton81]. The process requires the estimate of a "context function" which is a set of probability density functions for each possible neighbourhood configuration. Approximations of the context function made from a non-

context classification provided promising results, but accurate estimation proved to be a difficult problem.

Haralick set the problem of using context in a Bayesian framework [Haralick83] and examined the usefulness of Markov dependency within neighbourhoods. In an application of this work, Haralick and Joo provided a context formulation of the classification problem using a model in which the pattern vector associated with a particular pixel is only dependent on the true label (class) of that pixel and that the labels of pixels on any "path" (line) passing though that pixel have a Markov dependency on adjacent labels in the path [Haralick86]. This yields a straight-forward two pass algorithm involving an extra function trained to encode the dependency of pixels on their neighbours in four selected directions. To train this algorithm properly, ground truth data of a heterogeneous class structure is required.

Besag [Besag86] also provided a unified Bayesian framework for classification refinement, influenced by the work on image restoration using Markov random fields, especially the work of Geman and Geman, [Geman84]. Besag's algorithm, which he called "iterated conditional modes" incorporated empirically determined constants to use the estimate from one classification to affect the class conditional prior probabilities of the next. Similar algorithms were suggested by Kittler and Foglien [Kittler84] and Kiiveri and Campbell [Kiiveri86]. Kiiveri and Campbell give a conditional autoregressive version of classification improver for ML classification [Kiiveri91]. The use of Geman and Geman's ideas have also been applied to colour image classification by Wright [Wright89] and to remotely sensed image data classification by Hartt *et al.* [Hartt89] and Zhang *et al.* [Zhang90].

All of the methods described so far are forms of stochastic relaxation. Alternative approaches are possible using techniques from artificial intelligence. Fu discussed the use of syntactic reasoning to post-process the classified image [Fu76], but this has not achieved wide acceptance in remote sensing because of the nature of the data and the complexity of the processing required. Nichol applied region adjacency graphs to processing Landsat for the extraction of context information [Nichol90]. Wharton has described the use of context in an iterative knowledge-based classification system by incorporating heuristics that identify landcover which is "uniform", "border", "noise", or "outlier" in a classified image and incorporate these features into the next iteration [Wharton87].

1.4 Contributions of this thesis

This section briefly outlines each chapter and identifies the parts that I believe to be original contributions.

In Chapter 2 a method of covariance comparison due to Bogner [Bogner81a] is applied to the classification of pixel neighbourhoods. The chapter contains some new results about the distribution of the proposed metric, and an original implementation of the technique for use in images. A technique which is a hybrid of the neighbourhood classification method and the standard ML technique is presented and shown to give improved accuracy over ML.

Chapter 3 is about the use of multi-layer perceptrons for the classification of spectral data in a directly analogous manner to ML. The chapter includes a thorough exploration of the ability to control the behaviour of such a classification system through structural and parametric design. From the experimental work, the unavoidable conclusion is that it is necessary to repeat the training of a network from different starting points in order to be even reasonably sure of a sensible final trained network. In the second half of Chapter 3 some original insights are provided into the internal representation of a successfully trained network. Much of the work in this chapter was performed in parallel with other researchers, and their methods and results are discussed in the chapter.

Chapter 4 concerns methods of integrating spatial information into a neural network classifier. The aim is to allow a neural network to discover "texture". A number of schemes which try to distil neighbourhood structure in the input image are proposed and tested. The final network presented involves the unique structuring of input connections using clique concepts from Markov random field theory. It is hypothesized that this works because it assists the multi-layer perceptron to learn what inter-pixel spatial relationships are.

Chapter 5 proposes an original neural network based classifier which is designed to learn context. Some simple empirical results are given.

Chapter 6 concludes discussion of the new classification schemes with some comments on further applications and individual strengths and weaknesses. Some further work on the neural classifiers is proposed.

Chapter 2

Using Pattern Recognition by Observation Correlations to Investigate Spectral Structure.

Summary: *This chapter proposes a classifier in which each pixel is classified by calculating the covariance structure of the spectral bands over a spatial neighbourhood of the pixel, and then measuring the similarity of this structure to the covariance structures calculated on a set of reference regions. The similarity measuring algorithm, "pattern recognition by observation correlations" was developed originally for speech recognition. The use of this algorithm in remote sensing is justified and an implementation suitable multi-spectral data is described. An analysis of the algorithm incorporates some new results. When tested with Landsat data, the results are comparable with conventional ML classification, and a hybrid algorithm in which features derived using this covariance similarity metric and original data are merged is shown to be superior to either component classifier used alone.*

2.1 Inter-band information in multi-spectral images

In this chapter, we try to use the co-dependence of spectral measurements in the vicinity of the pixel to be classified as an extra source of information about the pixel. In order to characterize this property, we examine the covariance of the data in the neighbourhood of the pixel to be classified. The use of a classifier based on spectral covariance is of interest because the natural covariance in remotely sensed image data is at least in part class dependent. Schowengerdt, in describing Landsat MSS data ([Schowengerdt83], p160), ascribes the covariance to

1) the relatively low reflectance of vegetation in Landsat MSS bands 4 and 5, and relatively high reflectance in bands 6 and 7,
2) the shading effects caused by topographic slope and aspect, and
3) the overlap of spectral sensitivities between adjacent spectral bands, caused by practical limitations in filter construction.

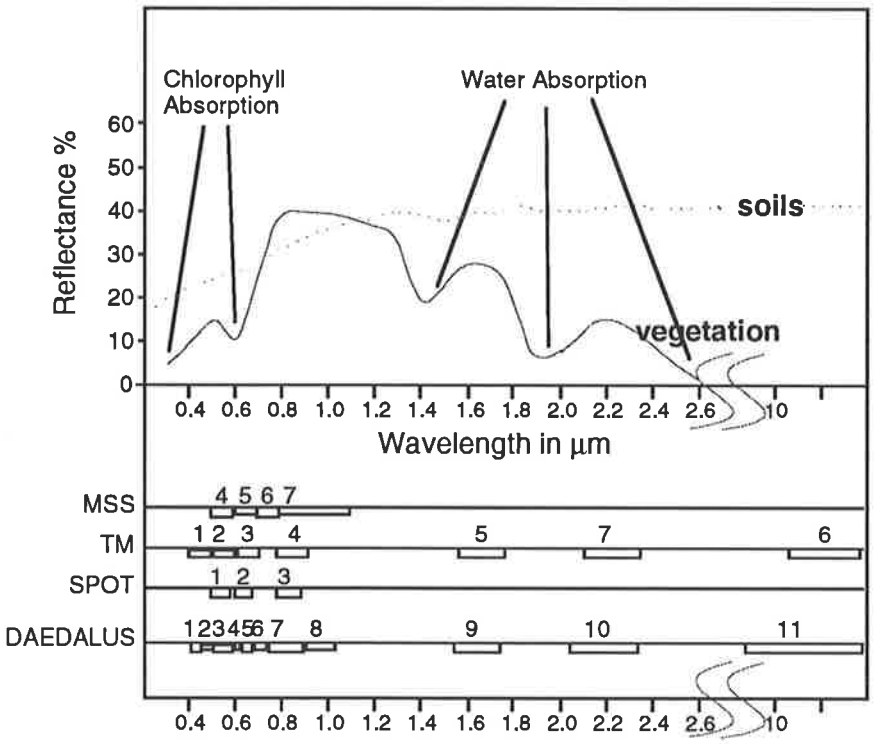


Figure 2.1: Reflectance vs wavelength plots for vegetation and soils.

The first mentioned effect is further demonstrated by figure 2.1, which shows typical vegetation and soil spectral reflectances compared to the spectral bands provided on three common remote sensing satellites and the Dædalus 1286 scanner. Kauth and Thomas [Kauth76] used these inter-band features in developing the so-called tasselled-cap transform, and more recently, similar spectral features have been used to look at sub-pixel classification and un-mixing [Settle91], [Pech86], [Smith90].

In a conventional ML classification scheme, mean vector $\boldsymbol{m}_r$ and covariance matrix $\boldsymbol{C}_r$ are the parameters used to characterize data that forms class ω_r. We visualize the data in N-dimensional (spectral) space as a fuzzy cloud whose centroid is described by $\boldsymbol{m}_r$ and whose shape and extent is described by $\boldsymbol{C}_r$ (figure 2.2). ML classification uses the fact that the probability of a particular point being in a class is dependent on its distance to the centroid, weighted by the expected spread of the class. A data point that is equidistant from two centroids will be placed in the class with the largest expected spread.

The main idea of this chapter is that, for a small neighbourhood of a pixel, it is possible to estimate a local covariance function that will describe a shape in N-dimensional space that is characteristic of a class. For this to be true it is necessary for the class have an inherent spatial/spectral structure which corresponds to a kind of multi-spectral texture. This latter assumption is shown to be well founded for real data.

Since primarily we wish to use covariance effects that are dependent on the landcover, i.e. effect (1) above, it will be necessary to find ways to avoid the undesirable aspects the other two factors. Effects due to limitations in spectral filter design can be partially eliminated by

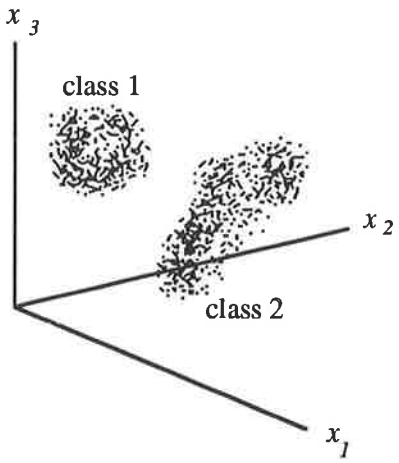


Figure 2.2: Scatter plot in spectral space for simulated data from 2 classes.

systematic removal as part of the classification scheme. Effects due to shading are both limiting because shading can mask data and useful because topology is often a source of extra information for classification. Conversely, under certain conditions, shading is not particularly difficult to remove. If we can model shading as a multiplicative effect which simply scales the data values, then, since a covariance-based measure does not require the value of the mean to be preserved, a suitable transformation on the image prior to classification can be used to remove the scaling. Unfortunately, shading can cause data values to be truncated by exceeding the lower limit of the digital number representation, so to make use of covariance it is necessary to limit application to domains having only minor amounts of shading.

Suitable algorithms for exploiting inter-(spectral-)band structure have been discussed in the speech signal processing literature. Li and Hughes observed that long-term correlation coefficients of speech exhibit intra-talker similarities and inter-talker differences and reported some results from an experimental study of three methods of comparing covariance matrices from continuous speech [Li74]. Bogner provided a theoretical basis for comparison of these covariance matrices and gave a procedure for making the comparison [Bogner81]. The procedure is based around an algorithm, "pattern recognition by observation correlations" (PROC), that gives a measure of how much the correlation between spectral bands in a test set of observations resembles the correlation between spectral bands in a reference set of observations. It was shown that the procedure could be applied when the means of samples are subject to confusion and hence classifiers which rely on the mean for discrimination are not generally useful.

Similarly, there are some situations in image processing where use of the ML classifiers is not relevant because the spectral radiance of the illumination is unknown or changing, and samples have texture. When trying to classify wide areas, the assumption that measurements of radiance are taken under conditions of constant illumination is found to be usually only approximately true for a single real image and is not true for a collection of images. Illumination in images taken by satellite over any extended period of time is affected by changes in sun angle caused by seasonal variation. Such changes are accompanied by a change

in the contributions made to the overall illumination by different spectral components.[1] It was hypothesized that the use of inter-band spectral structure would be useful in reducing the dependence on such spurious influences. Further, for a given set of sensors and where topographic effect is not extreme, it was hypothesized that the PROC metric could be used to create class signatures that would be transferable from one image to another and that could identify areas in cloud shadow..

The remainder of this chapter is devoted to recasting the PROC algorithm in a form suitable for image processing and then testing it with synthetic and real data. The resulting scheme showed that with some tuning the PROC generated features could be used for classification. A test of transferring of PROC class signatures from one image to another showed that the result produced reasonable results compared to useless results from the transferring of conventional class signatures for ML, but failed to give a system that was practically useful. Tests were also made using a scheme in which suitably transformed PROC features were combined with spectral features. This gave a classifier which utilized spatial neighbourhood data to give better classification accuracy than a ML system using spectral data alone. In forming a function akin to zero'th-order spectral texture, this technique has a lot in common with texture pre-processing methods.

2.2 PROC

2.2.1 A covariance comparison metric

Given set of multivariate data vectors, each vector of length M,

$$\mathbf{X} = \{x_1, x_2, \ldots x_{N_X}\}$$

and a set of reference data vectors, representative of the population

$$\mathbf{Y} = \{y_1, y_2, \ldots y_{N_Y}\}$$

with means m_X and m_Y and covariances C_X and C_Y, then it is relevant, under certain circumstances, to compare the covariance matrices C_X and C_Y for classification purposes. Pattern recognition by observation correlations (PROC) is a metric for measuring similarities between the covariances of sets of samples. The analysis is based on the paper by Bogner [Bogner81a].

It is assumed that the population variables are linear combinations of independent white random variables w_i of unit variance with added means m_i:

$$w`_i = w_i + m_i$$

Using the data generation model shown in figure 2.3, a member of the population may be described by:

$$\begin{aligned} y &= E\,\Lambda^{1/2}\,w` \\ &= E\,\Lambda^{1/2}\,(w + m) \end{aligned} \qquad (2.1)$$

---

[1] Altered spectral illumination at ground level is caused by variation in the amount of atmospheric absorption. This can be modelled effectively, given appropriate accompanying data (e.g. sun elevation), but such data is often unknown.

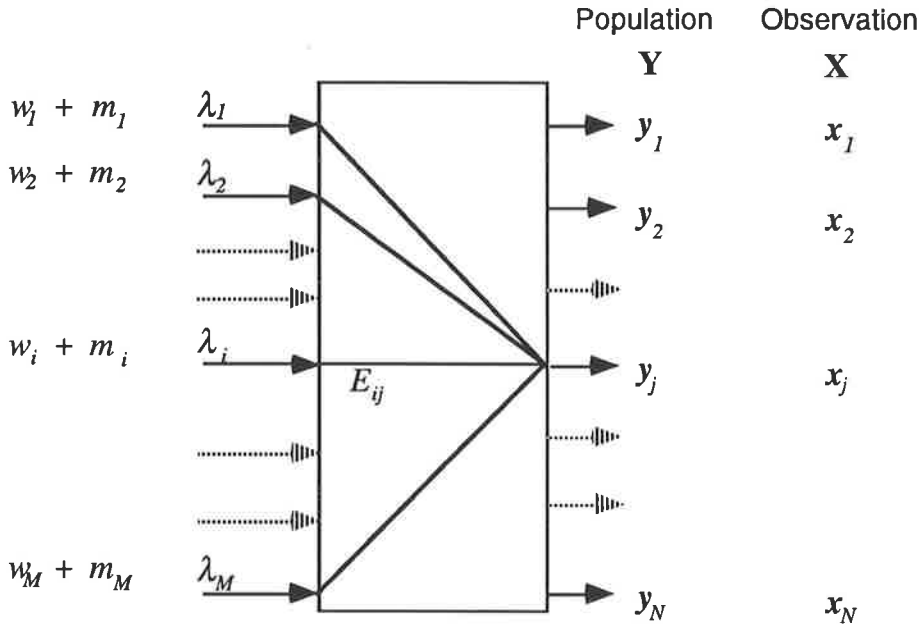


Figure 2.3: Model of the data generation process.

where $\boldsymbol{E}$ is the matrix of orthonormal eigenvectors of $\boldsymbol{C}_{\mathbf{Y}}$, and $\Lambda^{1/2}$ is the diagonal matrix with diagonal elements $= \sqrt{\lambda_i}$. The covariance matrix $\boldsymbol{C}_{\mathbf{Y}}$ is therefore given by:

$$\begin{aligned}\boldsymbol{C}_{\mathbf{Y}} &= \left\langle \left(\boldsymbol{E}\Lambda^{1/2}\,\boldsymbol{w}\right)\left(\boldsymbol{E}\Lambda^{1/2}\,\boldsymbol{w}\right)^{\mathrm{T}}\right\rangle \\ &= \left\langle \left(\boldsymbol{E}\Lambda^{1/2}\,\boldsymbol{w}\right)\left(\boldsymbol{w}^{\mathrm{T}}\left(\Lambda^{1/2}\right)^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}}\right)\right\rangle \\ &= \boldsymbol{E}\Lambda^{1/2}\,\boldsymbol{I}\,\left(\Lambda^{1/2}\right)^{\mathrm{T}}\boldsymbol{E}^{\mathrm{T}} \\ &= \boldsymbol{E}\Lambda\,\boldsymbol{E}^{\mathrm{T}} \qquad (2.2)\end{aligned}$$

which is the definition of eigenvectors and eigenvalues. ($\langle a \rangle$ signifies the expected value of a)

To compare covariance matrices, consider the matrix $\boldsymbol{C}_{\mathbf{X}}$, of elements

$$c_{\mathbf{X}ij} = \left\langle (x_i - \bar{x}_i)(x_j - \bar{x}_j) \right\rangle$$

estimated from observations, $\boldsymbol{x}$, and convert this into a form easily comparable with the identity covariance matrix $\boldsymbol{C}_{\mathbf{W}}$ of the independent $w`_i$ which has elements of the form

$$\begin{aligned}c_{\mathbf{W}ij} &= \left\langle \left(w`_i - \overline{w`_i}\right)\left(w`_j - \overline{w`_j}\right)\right\rangle \\ &= \langle w_i w_j \rangle\end{aligned}$$

If the observed **X** does indeed come from a model of the form of (2.2), then the corresponding inputs w_i may be estimated using the inverse of the transformation from **W** to **Y**, i.e.

$$\hat{w} = \left(\boldsymbol{E}\Lambda^{1/2}\right)^{-1}\boldsymbol{x} = \Lambda^{-1/2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{x}$$

Hence:

$$\begin{aligned}\boldsymbol{C}_{\hat{\mathbf{W}}} &= \left\langle \left[\Lambda^{-1/2}\,\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{X} - \bar{\boldsymbol{X}})\right]\left[\Lambda^{-1/2}\,\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{X} - \bar{\boldsymbol{X}})\right]^{\mathrm{T}}\right\rangle \\ &= \left\langle \Lambda^{-1/2}\,\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{X} - \bar{\boldsymbol{X}})(\boldsymbol{X} - \bar{\boldsymbol{X}})^{\mathrm{T}}\boldsymbol{E}\,\Lambda^{-1/2}\right\rangle \\ &= \left\langle \Lambda^{-1/2}\,\boldsymbol{E}^{\mathrm{T}}\boldsymbol{C}_{\mathbf{X}}\boldsymbol{E}\,\Lambda^{-1/2}\right\rangle \\ &= \left\langle \Lambda^{-1/2}\boldsymbol{C}_{\mathbf{Z}}\,\Lambda^{-1/2}\right\rangle \quad \text{where} \quad \boldsymbol{C}_{\mathbf{Z}} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{C}_{\mathbf{X}}\boldsymbol{E}\end{aligned}$$

$$c_{\hat{\mathbf{W}}\,ij} = \left[\sqrt{\lambda_i}\,\sqrt{\lambda_j}\right]^{-1} c_{\mathbf{Z}\,ij} \tag{2.3}$$

The elements of covariance matrix $\boldsymbol{C}_{\hat{\mathbf{W}}}$ have a variance of $\frac{1}{N}$ for $i \neq j$, and $\frac{2}{N}$ for $i = j$, and thus are ideally suited to incorporation in a Euclidean distance function:

$$D = \sum_{i=1}^{M} \frac{\left(\left(c_{\hat{\mathbf{W}}\,ij}\right) - 1\right)^2}{\left(\frac{2}{N}\right)} + \sum_{i=1}^{M} \sum_{j=1}^{i+1} \frac{\left(c_{\hat{\mathbf{W}}\,ij}\right)^2}{\left(\frac{1}{N}\right)} \tag{2.4}$$

which may be rewritten, using the property of matrix symmetry, as

$$D = \frac{N}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \left(c_{\hat{\mathbf{W}}\,ij} - \delta_{ij}\right)^2 \tag{2.5}$$

where δ_{ij} is the Dirac delta. D is further refined using some results on traces of matrices [Whatmough88]. Firstly we note that, from (2.2), we have

$$\boldsymbol{C}_{\mathbf{Y}} = \boldsymbol{E}\,\boldsymbol{\Lambda}\,\boldsymbol{E}^{\mathrm{T}}$$

which can be rearranged to give

$$\boldsymbol{I} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{C}_{\mathbf{Y}}\boldsymbol{E}\,\boldsymbol{\Lambda}^{-1/2}$$

where $\boldsymbol{I}$ is the identity matrix. By the definition of the trace of a matrix, we can reformulate D as

$$D = \frac{N}{2} \mathrm{Tr}\left\{\left[\boldsymbol{C}_{\hat{\mathbf{W}}} - \boldsymbol{I}\right]^2\right\} \tag{2.6}$$

$$= \frac{N}{2} \mathrm{Tr}\left\{\left[\boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{C}_{\mathbf{X}}\boldsymbol{E}\,\boldsymbol{\Lambda}^{-1/2} - \boldsymbol{I}\right]^2\right\} \tag{2.7}$$

$$= \frac{N}{2} \mathrm{Tr}\left\{\left[\boldsymbol{E}^{\mathrm{T}}\boldsymbol{\Lambda}^{-1}\boldsymbol{E}\left(\boldsymbol{C}_{\mathbf{X}} - \boldsymbol{C}_{\mathbf{Y}}\right)\right]^2\right\}$$

$$= \frac{N}{2} \mathrm{Tr}\left\{\left[\boldsymbol{C}_{\mathbf{Y}}^{-1}\left(\boldsymbol{C}_{\mathbf{X}} - \boldsymbol{C}_{\mathbf{Y}}\right)\right]^2\right\}$$

$$= \frac{N}{2} \mathrm{Tr}\left\{\left[\boldsymbol{C}_{\mathbf{Y}}^{-1}\boldsymbol{C}_{\mathbf{X}} - \boldsymbol{I}\right]^2\right\} \tag{2.8}$$

$$= \frac{N}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \left(\boldsymbol{C}_{\mathbf{Y}}^{-1}\boldsymbol{C}_{\mathbf{X}} - \boldsymbol{I}\right)_{ij} \left(\boldsymbol{C}_{\mathbf{Y}}^{-1}\boldsymbol{C}_{\mathbf{X}} - \boldsymbol{I}\right)_{ji} \tag{2.9}$$

Calculating D is relatively straight forward using (2.9). However, in practice it may well be just as quick and also as accurate to actually calculate the eigen-decomposition (2.2), and then calculate D directly from (2.7). In either case, it is desirable to add some small ε to $\boldsymbol{C}_{\mathbf{Y}}$ to avoid problems with near-zero eigenvalues

Bogner gave the expected value of distance D as

$$\langle D \rangle = \frac{\left[(M+1)M\right]}{2} \tag{2.10}$$

and ("plausibly rather than rigourously") the variance of D as

$$\left\langle D - \langle D \rangle^2 \right\rangle = M^2 + M \tag{2.11}$$

where M is the dimensionality of a sample. (This gives a standard deviation $\approx M$).

A more rigourous derivation of the mean and variance may be made by using a decomposition in sample number space, in a manner similar to that used to explore χ^2 distributions (suggested in a personal communication, [Whatmough90]). The derivation

proceeds by considering the generation of the bth variate of the transformed observation $\hat{w}_k$ over a number of samples, $k = 1 \ldots N$.[1]

Generate $(\hat{w}_{b1} \ldots \hat{w}_{bN})$ as

$$g_{b1}\frac{(1,1,\ldots,1)}{\sqrt{N}} + g_{b2}\frac{(1,-1,0,\ldots,0)}{\sqrt{2.1}} + g_{b3}\frac{(1,1,-2,0,\ldots,0)}{\sqrt{3.2}} + \ldots + g_{bN}\frac{(1,1,\ldots,1,-(N-1))}{\sqrt{N(N-1)}}$$

$$= \sum_{k=1}^{N} g_{bk} e_k \quad \text{for} \quad e_k = (e_{k1}, \ldots, e_{kN}) \tag{2.12}$$

Then the (b,c)th element of the covariance matrix is

$$c_{bc} = \frac{1}{N-1}\sum_{k=1}^{N} \hat{w}_{bk}\hat{w}_{ck} - \frac{1}{N(N-1)}\sum_{k=1}^{N} \hat{w}_{bk} \sum_{l=1}^{N} \hat{w}_{cl} \tag{2.13}$$

$$= \frac{1}{N-1}\sum_{k=1}^{N}\sum_{p=1}^{N}\sum_{q=1}^{N} g_{bp}e_{pk}g_{cq}e_{qk} - \frac{1}{N(N-1)}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{N}\sum_{q=1}^{N} g_{bp}e_{pk}g_{cq}e_{ql}$$

$$= \frac{1}{N-1}\sum_{p=1}^{N} g_{bp}g_{cp} - \frac{1}{N(N-1)} g_{b1}\sqrt{N}\, g_{c1}\sqrt{N}$$

$$= \frac{1}{N-1}\sum_{p>1} g_{bp}g_{cp} \tag{2.14}$$

Now,

$$\langle c_{bc} \rangle = \frac{1}{N-1}\sum_{p>1} \delta_{bc} = \delta_{cb} \tag{2.15}$$

Using Bogner definition of the "distance" D of $\boldsymbol{C}_{\hat{\mathbf{W}}}$ from $\boldsymbol{I}$ we observe that

$$\frac{2}{N} D = \sum_{b=1}^{N}\sum_{c=1}^{N} (c_{bc} - \delta_{bc})^2$$

$$= \sum_{b=1}^{N}\sum_{c=1}^{N}\left(\frac{1}{N-1}\sum_{p>1} g_{bp}g_{cp} - \delta_{bc}\right)^2$$

$$= \sum_{b=1}^{N}\sum_{c=1}^{N}\left\{\frac{1}{(N-1)^2}\sum_{p>1}\sum_{q>1} g_{bp}g_{cp}g_{bq}g_{cq} - \frac{2}{N-1}\delta_{bc}\sum_{p>1} g_{bp}g_{cp} + \delta_{bc}^2\right\}$$

which has the expected value

$$\left\langle \frac{2}{N} D \right\rangle = \sum_{b=1}^{N}\sum_{c=1}^{N}\left\{\frac{1}{(N-1)^2}\sum_{p>1}\sum_{q>1} \delta_{bc}^2 + \delta_{pq}^2 + \delta_{bc}\delta_{pq} - \frac{2}{N-1}\delta_{bc}\sum_{p>1} \delta_{bc} + \delta_{bc}^2\right\}$$

$$= \frac{1}{(N-1)^2}\left[M(N-1)^2 + M^2(N-1) + M(N-1)\right] - \frac{2}{(N-1)} M(N-1) + M$$

$$= \frac{M(M+1)}{(N-1)}$$

and hence

$$\langle D \rangle = \frac{M(M+1)}{2}\frac{N}{(N-1)} \tag{2.16}$$

which is Bogner's estimate corrected for bias.

To calculate the variance, we again consider $\frac{2}{N} D$, rather than D.

---

[1] To facilitate this manipulation, the order of the indices has been reversed. To avoid confusion, indices $b, c \ldots$ are used here rather than i, j.

$$\left(\frac{2}{N}D\right)^2 = \sum_{b=1}^{N}\sum_{c=1}^{N}\sum_{d=1}^{N}\sum_{e=1}^{N}\left\{\frac{1}{(N-1)^4}\sum_{p>1}\sum_{q>1}\sum_{r>1}\sum_{s>1} g_{bp}g_{cp}g_{bq}g_{cq}g_{br}g_{cr}g_{bs}g_{cs}\right.$$
$$- \frac{4}{(N-1)^3}\sum_{p>1}\sum_{q>1}\sum_{r>1} g_{bp}g_{cp}g_{bq}g_{cq}g_{br}g_{cr}\,\delta_{de}$$
$$+ \frac{4}{(N-1)^2}\sum_{p>1}\sum_{q>1} g_{bp}g_{cp}g_{bq}g_{cq}\delta_{bc}\,\delta_{de}$$
$$+ \frac{2}{(N-1)^2}\sum_{p>1}\sum_{q>1} g_{bp}g_{cp}g_{bq}g_{cq}\delta_{de}^2$$
$$\left. - \frac{4}{(N-1)}\sum_{p>1}\sum_{q>1} g_{bp}g_{cp}\delta_{de}^2 \;+\; \delta_{bc}^2\delta_{de}^2\right\} \qquad (2.17)$$

The expectation of the variance, collecting like terms and using $n = N - 1$, is given by

$$\left\langle\left(\frac{2}{N}D\right)^2\right\rangle = \frac{1}{n^4}\{M(M-1)(M-2)(M-3)[n(n-1).1 + n.1)]$$
$$+ M(M-1)(M-2)[n(n-1)(n-2).2 + n(n-1).14 + n.18]$$
$$+ M(M-1)[n(n-1)(n-2)(n-3).1 + n(n-1)(n-2).14 + n(n-1).65 + n.87]$$
$$+ M[n(n-1)(n-2)(n-3).1 + n(n-1)(n-2).18 + n(n-1).87 + n.105]\}$$
$$- \frac{4}{n^3}\{M(M-1)(M-2)[n(n-1).1 + n.1)]$$
$$+ M(M-1)[n(n-1)(n-2).1 + n(n-1).7 + n.9]$$
$$+ M[n(n-1)(n-2).1 + n(n-1).9 + n.15]\}$$
$$+ \frac{4}{n^2}\{M(M-1)[n(n-1).1 + n.1)] + M[n(n-1).1 + n.3)]\}$$
$$+ \frac{2}{n^2}M\{M(M-1)n.1 + M[n(n-1).1 + n.3)]\}$$
$$+ \frac{4}{n}M\{Mn\} + M^2$$
$$= M(M-1)(M-2)(M-3)\frac{n^2}{n^4} + M(M-1)(M-2)\left(\frac{2n^3+8n^2+8n}{n^4} - \frac{4n^2}{n^3} + \frac{2n}{n^2}\right)$$
$$+ M(M-1)\left[\frac{n^4+8n^3+34n^2+44n}{n^4} - \frac{4(n^3+6n^2+4n)}{n^3}\right.$$
$$\left. + \frac{4n^2}{n^2} + \frac{2.2n}{n^2} + \frac{2(n^2+2n)}{n^2} - \frac{4n}{n} + 1\right]$$
$$+ M\left[\frac{n^4+12n^3+44n^2+48n}{n^4} - \frac{4(n^3+6n^2+8n)}{n^3}\right.$$
$$\left. + \frac{4n^2}{n^2} + \frac{2.2n}{n^2} + \frac{2(n^2+2n)}{n^2} - \frac{4n}{n} + 1\right]$$
$$= M(M-1)(M-2)(M-3)\frac{1}{n^2} + M(M-1)(M-2).\left(\frac{8}{n^2} + \frac{8}{n^3}\right)$$
$$+ M(M-1)\left(\frac{18}{n^2} + \frac{44}{n^3}\right) + M\left(\frac{12}{n^2} + \frac{48}{n^3}\right)$$

$$\left\langle\left(\frac{2}{N}D\right)^2\right\rangle = \frac{(M^4+2M^3+M^2)}{n^2} \qquad (2.18)$$

$$\mathrm{var}\left(\frac{2}{N}D\right) = \frac{4M(M+1)}{(N-1)^2} + \frac{(8M^3+20M^2+20M)}{(N-1)^3}$$
$$= \frac{4M(M+1)}{(N-1)^2} + \frac{4M(4M^2+5M+5)}{(N-1)^3}$$

$$\text{var}(D) = M(M+1)\left(\frac{N}{N-1}\right)^2 \left\{1 + \left(\frac{1}{N-1}\right)\left[2M+3+\left(\frac{2}{M+1}\right)\right]\right\} \qquad (2.19)$$

$$\approx M(M+1) \quad \text{for} \quad N >> 2M+4$$

Table 2.1 shows the expected values for D calculated using both the corrected and uncorrected formulae. As expected for $N = 100$, there is little difference.

Table 2.1: Theoretical expected values of D and variance(D) for $N = 100$

Bands	Exp. Value (Corrected for Bias)	Exp. Variance (Corrected for Bias)	Exp. Value	Exp. variance
1	1.01	2.16	1.00	2.00
2	3.03	6.60	3.00	6.00
3	6.06	13.42	6.00	12.00
4	10.10	22.76	10.00	20.00
5	15.15	34.73	15.00	30.00
6	21.21	49.47	21.00	42.00
7	28.28	67.09	28.00	56.00
8	36.36	87.73	36.00	72.00
9	45.46	111.49	45.00	90.00
10	55.56	138.51	55.00	110.00
11	66.67	168.92	66.00	132.00

The distribution of D is related to the Wishart distribution which is in turn related to the chi-squared distribution. If $\mathbf{W} = \{w_1, w_2, \ldots w_N\}$ is sampled from a multivariate normal distribution with zero mean and covariance $\boldsymbol{C}_\mathbf{W} = \frac{1}{N-1}\mathbf{W}\mathbf{W}^T$, we write:

$$\mathbf{W} \sim \mathcal{N}(0, \boldsymbol{C}_\mathbf{W}),$$

Then the matrix $\widehat{\mathbf{W}} = \boldsymbol{C}_\mathbf{Y}^{-1/2}\mathbf{W}$ is sampled from $\mathcal{N}(0, \boldsymbol{I})$, so that $\mathbf{W}\widehat{\mathbf{W}}^T$ has a Wishart distribution, written

$$\mathbf{W}\widehat{\mathbf{W}}^T \sim \mathcal{W}_M(N, \boldsymbol{I}) \qquad (2.20)$$

The closed form expression for the distribution of

$$\boldsymbol{C}_{\widehat{\mathbf{W}}} = \frac{1}{N-1}\mathbf{W}\widehat{\mathbf{W}}^T \sim \mathcal{W}_M\left(N, \frac{1}{N-1}\boldsymbol{I}\right) \qquad (2.21)$$

is given by

$$p(C_{\widehat{\mathbf{W}}}) = \frac{|C_{\widehat{\mathbf{W}}}|^{(N-M-1)/2}\exp\left(\frac{1-N}{2}\text{tr}\,(C_{\widehat{\mathbf{W}}})\right)}{2^{NM/2}\pi^{M(M-1)/4}(N-1)^{MN/2}\prod_{i=1}^{M}\Gamma\left(\frac{1}{2}(N+1-i)\right)} \qquad (2.22)$$

if $N > M$ and $C_{\widehat{\mathbf{W}}}$ is positive definite ([Mardia79], p85). In principle, the distribution of derived quantities, in particular D, could be derived from (2.21) (see [Muirhead82]). However,

this process is tedious, complex and of little practical value. It is more useful to consider whether some simple function, $f(D)$, might have a distribution with moments approaching those of the normal distribution in some limit, and hence $f(D)$ could reasonably treated as normal. The probability density function of a random variable is uniquely determined if all its moments are known ([Papoulis84], p116, moment theorem).

The estimation of the moments of $f(D)$ using simulated random data is straight forward and not particularly compute intensive. Tables 2.8 to 2.13 show the mean, the variance and next 6 (normalized) moments of a simulated distribution of $f(D)$ for $M = 1$ to 11 bands, with 10x10 neighbourhoods. The nearest approach to normality is when $f(D) = \sqrt[4.5]{D}$. but as was suggested by previous work with speech data, $\log(D)$ can be seen to be a useful approximation (cf. tables 2.9 and 2.12 for number of bands, $M = 4$.)

2.2.2 Application to remotely sensed images

In conventional schemes of multi-spectral image classification, each pixel is classified on the basis of its pattern vector, into one of R classes. The classes are defined by parameters estimated by observing the pattern vectors of a reference sets $\mathbf{Y}_1$, $\mathbf{Y}_2$, ... $\mathbf{Y}_r$, ... $\mathbf{Y}_R$, where the reference sets are abstracted from the complete image.

To apply PROC to image data, the distance D from a reference set is regarded as being a characteristic of a pixel and its neighbourhood, i.e. radiance vectors for a collection of pixels in some defined sub-area of the image are considered to be multiple observations of the same data. We define a neighbourhood system as

$$\eta_{kl} = \{(i_1,j_1), (i_2,j_2), \ldots (i_{N_\eta-1},j_{N_\eta-1}), (k,l) : (i_n,j_n) \subset L \} \tag{2.23}$$

and η_{kl} is the neighbourhood of (k,l) on a finite lattice L, such that

$$(i,j) \in \eta_{kl} \Leftrightarrow (k,l) \in \eta_{ij}$$

and N_η is the number of pixels in the neighbourhood.

The most general neighbourhood is not inherently structured, or even necessarily connected, but it is convenient to use the hierarchical sequence of neighbourhood systems that is frequently used in image processing (figure 2.4). Neighbourhood systems are of interest in modelling images as Markov random fields [Derin89].

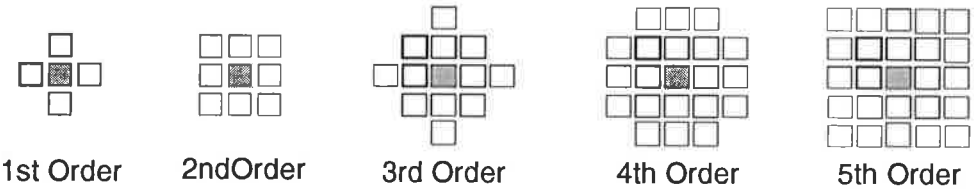


Figure 2.4: Hierarchical neighbourhood systems.

A convenient notation for the neighbourhood in an image X is

$$X_{ij}^{\eta} = (x_{kl} \mid (k,l) \in \eta_{ij}) \tag{2.24}$$

In the image processing implementation, the set **X** in section 2.2.1 is replaced by X^{η}, the data from the neighbourhood of the pixel to be classified. PROC features are the distances $D(C_{X^{\eta}}, C_{Y_r})$, $r = 1, \ R$.

When comparing class covariances, the class means of the data are not used and hence preservation of the means is not necessary. We would like to remove the spectral band biases caused by having different levels of illumination in different bands. These can be the cause of misclassification as these differences can arise from variations of transmission or variations in the spectral radiance of the effective illuminator (sun plus sky) rather than from differences in reflectors (land covers). This process also has the potential to remove biases caused by transmission through haze and mild cases of shading, as discussed above.

In real image data, for each pixel with pattern vector x_{kl}, the value $x_{kl,i}$ is a digital number related to the radiance measured observed by a sensor in spectral band i by

$$x_{kl,i} = S_i I_{kl,i} r_{kl,i} + S_i A_i + S_i O_i \tag{2.25}$$

where S is sensor gain, $I_{kl,i}$ is band illumination, $r_{kl,i}$ is ground reflectance, A_i is atmospheric path radiance, and O_i is sensor calibration offset [Kowalik83]. The subscripts k and l have been retained here to indicate that some quantities may be functions of position. The approximately constant offset term $(S_i A_i + S_i O_i)$ does not affect classification schemes using relative radiance, e.g. ML, but must be removed for any scheme which uses absolute data, for example, spectral band ratio-ing.

The constant offset term $(S_i A_i + S_i O_i)$ in equation (2.25) can be removed for MSS data by subtracting an offset found using the deep water approximation to zero. (Where an image contains no water, it would be necessary to use a regression method such as the regression intersection method [Crippen87].) When the offset term is removed, the radiance is given by

$$x_{kl,i} = S_i I_{kl,i} r_{kl,i}$$

The term $S_i I_{kl,i}$ is slowly varying in comparison to $r_{kl,i}$ such that it may be considered constant with respect to k and l for some small neighbourhood of (k,l). We identify $H_i = S_i I_{kl,i}$ as the unwanted "gain" factor which we want to remove. Then if we use $\log(x)$ we can form the covariance matrix from elements c_{ij} given by:

$$\begin{aligned} c_{ij} &= \left\langle \left(\log x_{kl,i} - \overline{\log x_i}\right)\left(\log x_{kl,j} - \overline{\log x_j}\right)\right\rangle \\ &= \left\langle \left(\log H_i r_i + \overline{\log H_i r_i}\right)\left(\log H_j r_j + \overline{\log H_j r_j}\right)\right\rangle \\ &= \left\langle \left(\log r_i + \overline{\log r_i}\right)\left(\log r_j + \overline{\log r_j}\right)\right\rangle \end{aligned} \tag{2.26}$$

where $\overline{\log r_{ii}}$ is the mean value of $\log r_i$ in the ith dimension (band), over a neighbourhood of (k,l).

Drawing together the pre-processing and the PROC method, it is clear that the value of data at location (k,l) is being modelled by

$$\log x_{kl,i} = m_i + w_{kl,i}$$

where m is the mean value which is not of interest in generating PROC features and $w_{ij,k}$ is a zero mean random variable with significant covariance structure with respect to the correlation

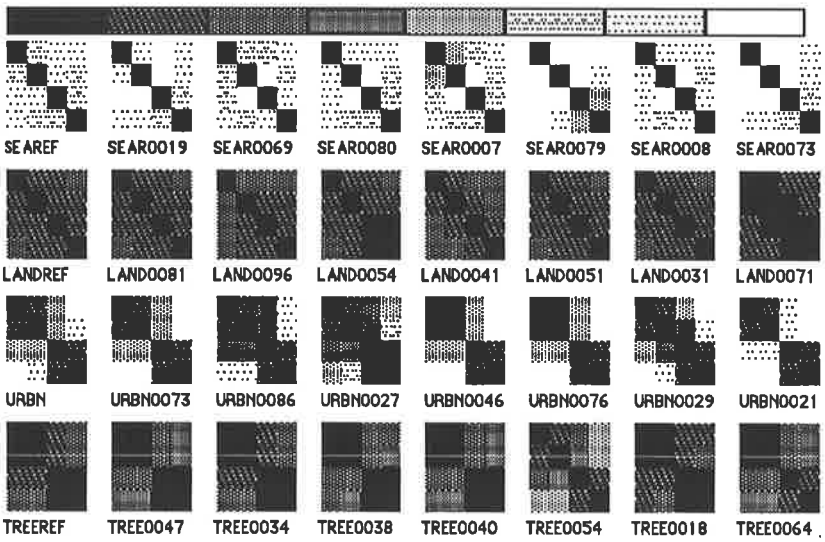


Figure 2.5: Shading diagrams representing correlation coefficients.

between bands (i.e. i) which contains information about important real variations in the actual data.

2.2.3 Comparison of sub-areas – feasibility of the method

Before classifying complete images, some tests were performed using data abstracted from MSS scenes to establish that results were reasonable.

Large square areas of an image, judged to be homogeneous, were designated as reference areas. Each reference area was chosen to be representative of a particular ground-cover class, using the same criterion used for selection of a training set for ML classification. From within each reference area 100 test areas of size 9x9 pixels were also defined. The matrices of covariances were formed for each set and values for D were calculated to measure the similarity between the covariance matrices. For each reference set, values of D were computed to compare the complete reference area (reference set) and all defined subsets.

Following the line of investigation used in speech processing, shading diagrams were used as a "first approximation" guide to the applicability of the method to images. The type of shading diagram used here represents a normalized covariance matrix by a block of M by M squares, where M is the dimension of the data. A square at position (i, j) has a grey shade proportional to the normalized value of c_{ij}. Covariance matrices were normalized by dividing elements c_{ij} by $\sqrt{(c_{ii}c_{jj})}$, which transforms them into correlation coefficients.

The shading diagrams for sets generated from four reference areas are shown in figure 2.5. This figure shows the (normalized) covariance matrix for the whole reference area in the first diagram of each horizontal sequence, followed by the 7 samples with the smallest value of D. (Black indicates a value +1, white indicates a value of -1). There are obvious similarities between diagrams of sets taken from the same reference area, and obvious differences between sets taken from different reference areas. These observations parallel results from using the algorithm with speech data where it was found that utterances from a particular talker give rise

to similar shading diagrams [Bogner81]. To examine the properties further in comparison to the speech results, it was appropriate to investigate the properties of log(D). In the work done in speech, the frequency distribution for log(D) was found to be substantially Gaussian, and in section 2.2.1, this was shown to give a reasonable approximation to the Gaussian. (The use of a transformation using the 4.5th root would have made comparison with the speech results difficult and was a more complicated transformation).

Figure 2.6 shows the graphical representation of the results for the reference area of "urban". Values of log(D) are accumulated from the right for sub-areas taken from "urban", and from the left for sub-areas taken from "sea", "trees" and "land". Hence the "urban" curve shows number of samples with values greater than the corresponding value of log(D), and the other curves show number of values less than log(D). From figure 2.6 it can be seen that by choosing a threshold of log(D)=1.8, better than 80% of sub-areas from "urban" would be classified correctly.

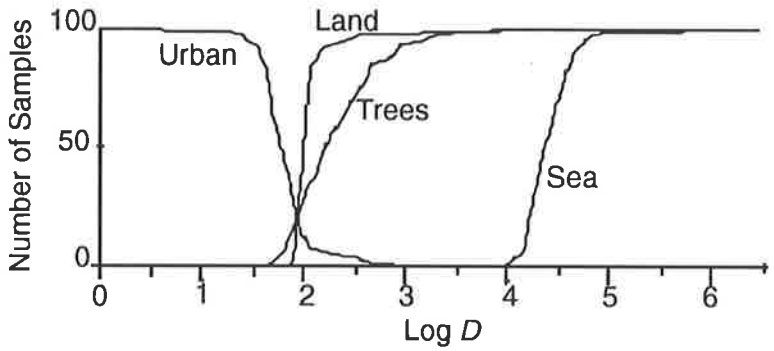


Figure 2.6: Cumulative distribution of D for test areas compared to "urban".

The expected value and variance of D (equations 2.10, 2.11) are dependent essentially on the number of dimensions of the data. With $M = 4$ the expected value of $D \approx 10$, and the expected variance of D is 20 (gives standard deviation = 3.5). Clearly this is only marginally viable, but for satellite image classification the possibility of classifying 4 band images is of considerable interest.

Table 2.2 presents the average values of D when comparing covariances of sub-areas from a reference area to the covariance of a complete reference area. The average value has been calculated using the geometric mean, which is more indicative of the central tendency of D because of the Gaussian nature of the log(D) distribution. The average value of D is 4 to 7 times the expected value. This could be explained by the small number of bands and the spatial overlap of field of view of the sensors when measuring individual pixel data. Here, too, there are parallels with the speech work [Bogner81]. The mean self distances measured from examples of speech were of the order of 10 times greater than the calculated theoretical values. This was explained by the time dependence of the pattern features being estimated, where estimates were made every 10mS for speech with a phoneme rate of approximately 10 per second.

Table 2.2: Average of D over subsets of a reference area

Ref. area	Sub-area parent names			
	SEA	LAND	URBAN	TREES
SEA	42	238	786	3516
LAND	34938	65	863	1165
URBAN	24751	109	63	193
TREES	13062	116	93	51

2.2.4 Moving window implementation

The PROC algorithm for image classification was implemented as a pixel-by-pixel classifier using similar techniques to those used for texture analysis and filtering. Ordered neighbourhoods are used as defined above. The image is effectively scanned by a moving window of a shape determined by the neighbourhood's order. When the neighbourhood is square, this method has previously been called a "box car" technique.

If p is the length of the longest row or column in the neighbourhood, pixels closer to the edge of the image than $p/2$, cannot be classified and appear as an unclassified border in the output image.

Processing proceeds as follows (using some constructs from *pascal* notation):

Replace the values in each pixel's vector by their log : $x_{kl,r} \rightarrow \log(x_{kl,r})$ $r = 1 \ldots R$

For each of R reference sets **do**

Calculate covariance $\boldsymbol{C}_{\mathbf{Y}r}$ and the inverse $\boldsymbol{C}_{\mathbf{Y}r}^{-1}$

For each of the pixels in the image, but not in the border **do**

Begin Calculate the spectral covariance $\boldsymbol{C}_{\mathbf{X}}$ of the pixels contained in the moving window centred on the current pixel,

For each of R classes **do**

Calculate D_r

Compare D_r's; select and record class of smallest

end {loop to process next pixel}

{end of algorithm}

The classification steps are shown diagrammatically in figure 2.7.

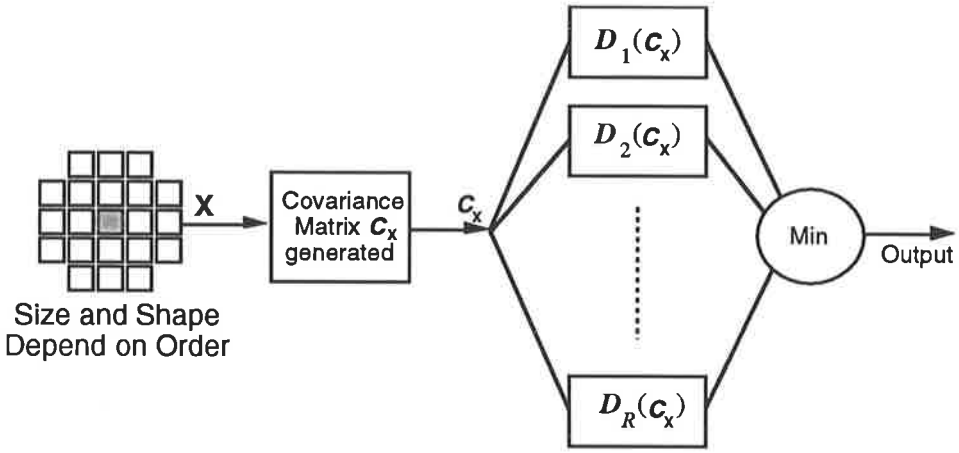


Figure 2.7: PROC implementation for image classification.

The results of varying window size were tested using synthetic images, generated by using the model in 2.2.1 with parameters taken from real data. Examples were tested with real data was taken from 4-band multi-spectral images. A test image was formed from 4 synthetic 20x20 sub-images, one from each class. Accuracy was measured on the central 10x10 square in each sub-image. Mis-classifications occurred at random throughout the image for small window sizes. For window sizes of 7x7 (9th order) or greater, mis-classified pixels were generally at edges. Increasing order improved accuracy of classification markedly (figure 2.8), helped by increasing sample sizes for the calculation of covariance.

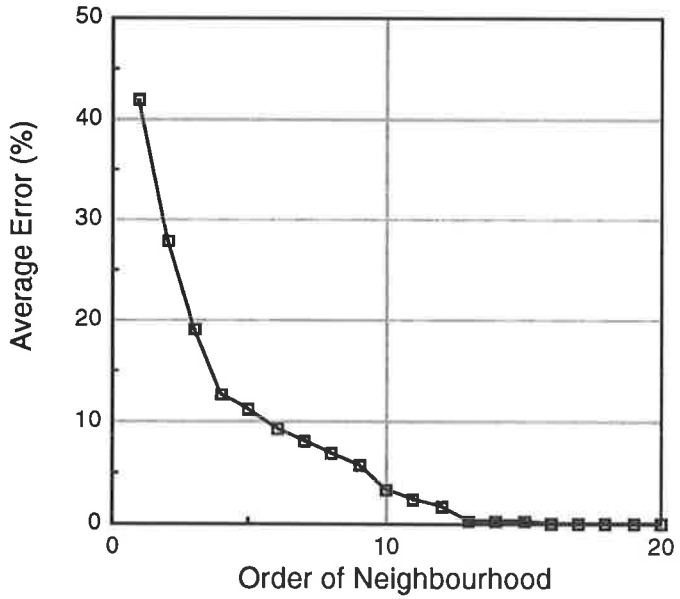


Figure 2.8: Av. classification error vs. order neighbourhood for 4-band test images.

A number of unavoidable artifacts are introduced by using a moving window implementation of the algorithm. A collection of pixels used to characterize a single pixel necessarily reduces resolution – a common side-effect of analysis using texture measures. Window shape effects the shape of boundaries between classes and by its nature covariance at edges takes on unusual values. While these limitations would be difficult if the goal of

classification was to identify features of the same size as a resolution element in the image, for wide area classification of remotely sensed images, these artifacts are not so important.

2.3 Examples of image classification using PROC

Experiments were carried out to verify that the PROC algorithm with a 9th order window could be used as a classifier, and to test if PROC could make sensible classifications under the conditions where conventional classifiers are not usable: classifying areas in cloud shadow, classifying areas under thin haze[1] and classifying a second image without retraining.

Section 2.3.1 discusses the use of PROC to classify known reference areas taken from the test images. For these reference areas, the ground truth is known accurately and it is possible to make a good assessment of the accuracy of classification. Section 2.3.2 discusses the classification of complete images, presenting a more synoptic view of the abilities of the algorithm and analysing some difficulties that arise in the classification of real images. This separation of different kinds of algorithm assessment avoids the requirement for a pixel-by-pixel hand-classification of a large image in order to gain an overall assessment of the algorithm.

Algorithms were initially tested using Landsat MSS data. A 512 line by 640 column subset of the "Adelaide" scene was chosen to include some diverse, easily ground-truthed land-cover classes. The experiments used two sets of data, which were for the same geographic area but collected on different dates. A subset from an image taken in March, 1985[2] is referred to as the primary image in this chapter (figure 2.9) and a subset of an image taken in October 1984[3] is referred to as the secondary image (figure 2.10).

Reference areas were selected from the primary image using local knowledge of the area and some photo-interpretation of the image. These areas were chosen to be as homogeneous as possible. The reference areas were used to generate statistics to characterize 5 classes loosely named "water", "cloud", "urban", "trees", and "farm". These categories were chosen to be representative of the kind of classes that would be appropriate in wide-area classification (figure 2.9). No reference areas were selected in the secondary image, which was only used to test the possibility of transferring class parameters from one image to another. Classification was performed on data from both images using the class statistics generated from the primary image. For comparison, classifications were performed using both the PROC classifier and the standard ML classifier.

2.3.1 Classification of image subsets to assess potential accuracy

For tests described in this section reference areas were divided into two sections, one used for training and the other used for measuring accuracy. Results of classifications are presented in the form of contingency tables (tables 2.3 – 2.7) showing what percentage of true members

[1] Where training areas are not in shadow or under haze.

[2] Label is :- Lat = 3439, Long = 13859, Date = 291084, GMT = 3, Sun Az = 382, Sun El = n.a.

[3] Label is :- Lat = 3439, Long = 13901, Date = 220385, GMT = 4, Sun Az = 326, Sun El = n.a.

of a class were classified as members of each of the possible choices of class. (An ideal table would show 100% on the diagonal and 0% everywhere else.)

Initially, the PROC method produced poor results when categories were assigned simply by selecting the maximum covariance similarity as indicated by the minimum D-value, following the scheme shown in figure 2.7. On examination, D-values for a class appeared to incorporate a multiplicative bias, possibly due to the relative size of spatial features in the class. The D-value for a class is biased when data from a that class occurs in spatial clusters of a size less than or similar to the moving window size. Empirically, another source of bias was found to be the size of the training sets, which for practical reasons were not all equal.

To defeat this bias, weighting factors were derived for the classes by examining self classification D-values of moving-window-sized sub-samples of the training sets. The selection of the weights by measuring a property of the training set incorporates some measure of relative textural primitive feature size into the classification process. The weights were used to scale D-values prior to selection of a maximum during classification. With these problems ameliorated, the examination of data showed that there were occasions when PROC classifier was clearly better than the standard ML, while there are many occasions when it is more sensible to use class mean based classification of the ML.

To combine the classification power for PROC and ML, a classifier was developed in which each pixel's feature vector of radiance values was augmented with elements representing D_r's and then these feature vectors were classified using the standard ML scheme. To use such a vector in a meaningful way under a ML classification scheme it is desirable that the form of the distribution of each of the components of the vector be the same. In the standard ML scheme, it is assumed that the vector is a sample from a multivariate normal (Gaussian) distribution. This is not the case when some of the components come from a quite different distribution. However, as discussed in section 2.2.1, transforming the D_r-values using a log function or more correctly the 4.5th root function will give a distribution that is substantially Gaussian.

Hence, the hybrid scheme comprised

1) Generation of an image containing PROC features measured at each pixel.
2) Transformation of features into a usable range using the 4.5th root.
3) Augmentation of the pattern vector of radiance measurements for each pixel by including the transformed D-values, creating a hybrid image.
4) ML classification on the new multi-dimensional data sets to produce a class map.

This scheme gave improved results over both ML and PROC.

Table 2.3: Contingency table for primary image PROC classifications

True class	Class assigned (% true class)				
	SEA	CLOUD	URBAN	TREES	FARM
SEA	100.0	0.0	0.0	0.0	0.0
CLOUD	0.0	98.5	0.0	1.5	0.0
URBAN	3.3	0.0	96.5	0.3	0.0
TREES	0.0	0.0	0.3	99.8	0.0
FARM	0.0	0.0	23.5	0.5	76.0

Table 2.4: Contingency table for primary image ML classifications for comparison

True class	Class assigned (% true class)				
	SEA	CLOUD	URBAN	TREES	FARM
SEA	100.0	0.0	0.0	0.0	0.0
CLOUD	0.0	100.0	0.0	0.0	0.0
URBAN	0.0	4.2	88.2	0.0	7.6
TREES	0.0	0.0	0.0	96.5	3.5
FARM	0.0	0.0	4.9	4.2	91.0

Table 2.5: Contingency table for primary image classifications using combined PROC/ML

True class	Class assigned (% true class)				
	SEA	CLOUD	URBAN	TREES	FARM
SEA	100.0	0.0	0.0	0.0	0.0
WATER	0.0	100.0	0.0	0.0	0.0
URBAN	0.0	0.0	95.1	0.0	4.9
TREES	0.0	0.0	0.0	100.0	0.0
FARM	0.0	0.0	0.7	0.0	99.3

2.3.2 Classification of complete images

To evaluate the complete image classifications presented in figure 2.11 it is necessary to be familiar with some of the geography of the area where the data was collected. Here follows a brief geographic introduction to figures 2.9 and 2.10 which are images of the city of Adelaide

and environs. These images are Landsat MSS printed in the conventional false colours that emulate infrared film. Vegetation appears red because it is a good reflector in the infrared.

The most obvious feature in the images is the sea to the west of the city. The port for the city is formed by the hook shaped landform at the top left of the image, although the harbour and docks are mostly situated upstream (lower in the picture) on the "Port River". The river is entered by shipping from the sea heading north, then turns through east and heads south. The main docks occur at an obvious man-made right angle in the river. The island in the hook is "Torrens Island" which is mostly covered by mangroves, hence the intense red in the image. There is an adjoining section of swampy mainland also covered in mangroves. The intense blue colour in man-made shapes near the mangroves is from salt evaporation pans. The darker coloured man made shapes north of the mangroves are sewerage treatment ponds.

The city of Adelaide was originally planned to occupy a "square mile" surrounded by park lands. This is the blue shape left of centre image surrounded by red. There is a lesser square, adjacent and to the north of the city square mile also edged by park. Urban development surrounds the business district in typical Australian style, with 1/4 acre blocks containing a house, some vegetation and an occasional swimming pool. Industrial developments diverge only mildly from this urban land use pattern, with some obvious exceptions where there is a large factory.

The urban development is constrained (by legislation) from spreading into the foothills of a low north-south range to the east of the city. On the far side of the range is open farmland used for farming - mostly grazing. Within the range there are small farms, a number of state forests and some forest reserves surrounding reservoirs (see especially the "Crawford forest" at the top centre of the image and a forest reserve surrounding a reservoir south of the Crawford forest).

The reference areas used were a section of sea, a cloud, a section of urban development north of the city centre, the Crawford forest, and a section of open country east of the range. Remote sensing methodology for classifying multi-spectral images using ML requires a far more diverse set of training areas to ensure that the training set contains examples that can, as near as possible, approximate all expected values of data to be classified. In this case, simple restricted training areas were used so that the new methods could be compared to the established method on the basis of the same training sets.

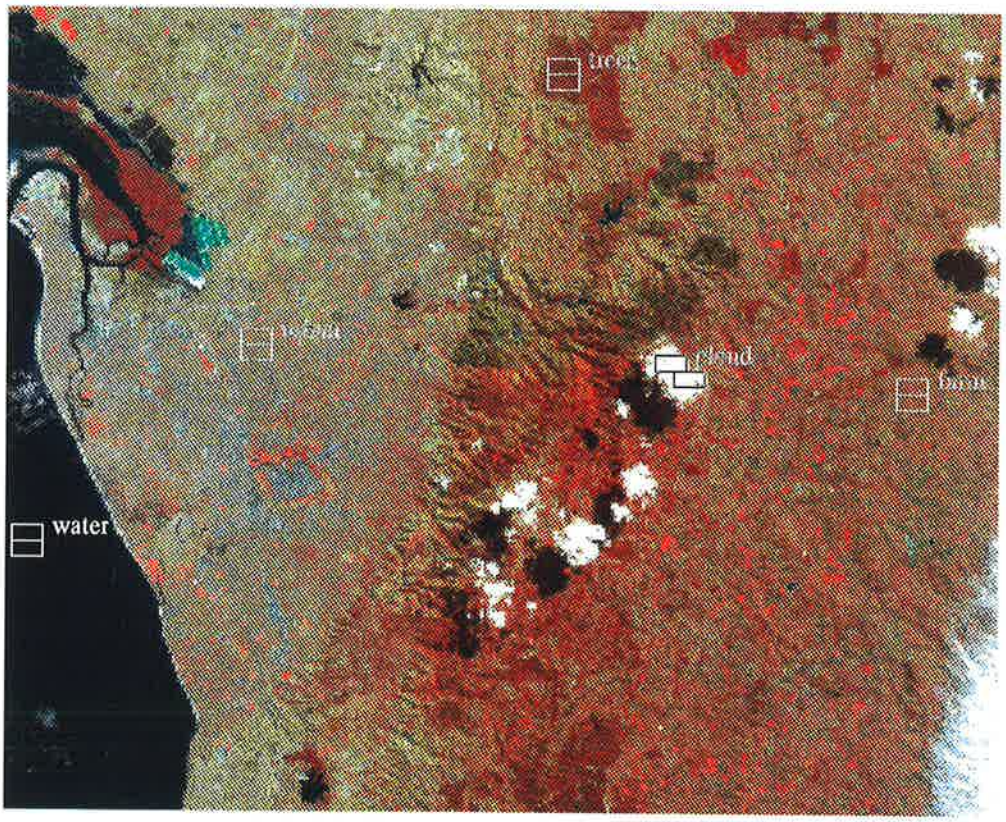


Figure 2.9: The primary image, with reference areas outlined

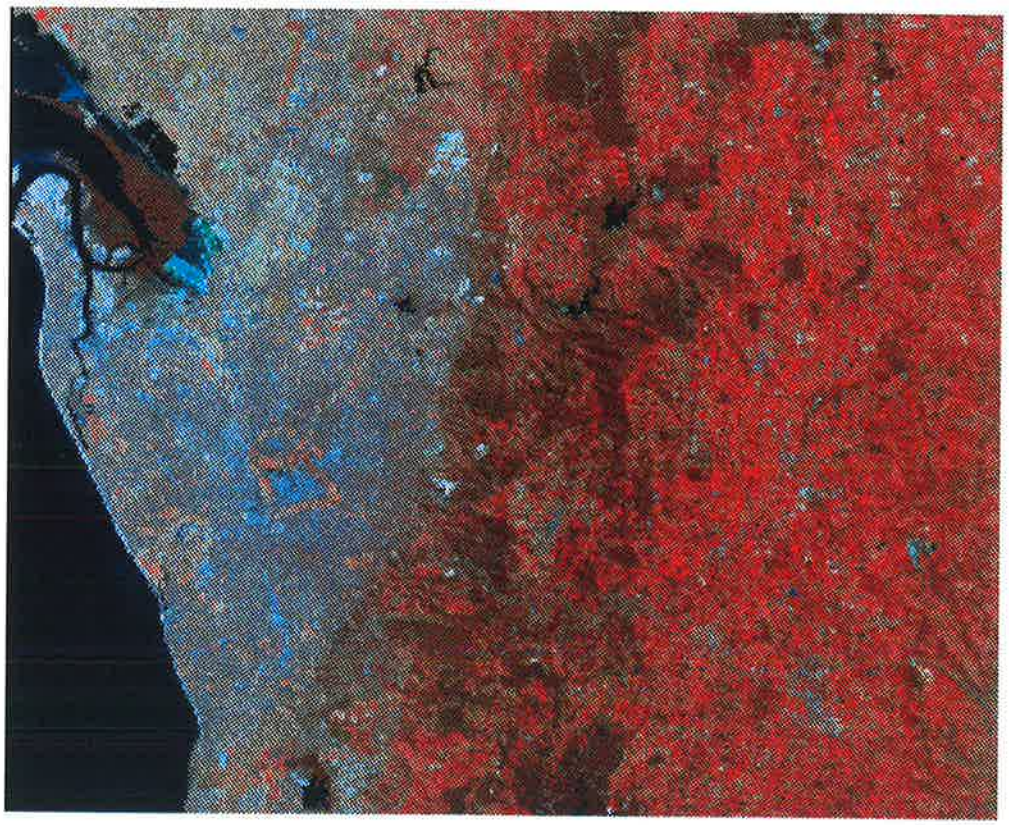


Figure 2.10: The secondary image

Figure 2.11 shows the classifications of the primary and secondary images using the two methods based on PROC and using the standard ML scheme. As expected, the ML classification (bottom row) is a reasonable classification of the Adelaide scene, but with some mis-classifications of typical problem areas. For example, shallow water in the tidal inlet (adjacent to the "Port River") in the upper left of the figure is misclassified. Cloud shadows (which appear below and to the left of the small clouds in the middle of the scene) have been classified surprisingly well. The class "urban" contains some diverse spectral data (i.e. with texture) and is particularly difficult to characterize with the parameters available to the ML scheme, especially because of its reliance on the mean radiance value as its prime feature. Because of its diversity, the class "urban" has become a class into which problem pixels have been placed.

The PROC classification (using weighted *D*-values) agrees with the ML classification on the whole, even though it does not use a pixel's radiance values, but only properties of its neighbourhood. The resolution of the classmap appears coarser, which is a side effect of the moving window implementation. The classification differs in the Port River area, where the water has been the dominant class. Some reservoirs missed under ML have also been highlighted at the expense of classifying peripheral pixels incorrectly. In the PROC classification, the catch-all class for problem pixels appears to be water; e.g. water (mis-)classification on the eastern side of the green centre strip .

As expected the results of classifying the secondary image using maximum likelihood classification were not useful; for example, 142,060 out of the total of 326,529 pixels were classified as cloud, and there is no cloud at all in the scene. The PROC (weighted *D*-value) classification also produced some mis-classifications, but only of a similar nature to those errors in the original classification task.

Tables 2.6 and 2.7 give an approximation of the contingency matrices for classification of the whole secondary image by the weighted PROC and ML methods. This table summarizes the comparison to the classmaps produced by each method and the "true" classes of each pixel. The image size was 640 x 512 pixels. 6876 pixels were not classified by PROC because of edge effects and are not included in the table.

Table 2.6: Contingency table for secondary image PROC classifications

Est. true class	Class assigned by PROC (% true class)				
	SEA	CLOUD	URBAN	TREES	FARM
SEA	99.1	0.0	0.0	0.5	0.4
CLOUD	n.a.	n.a.	n.a.	n.a.	n.a.
URBAN	9.9	0.3	54.0	18.4	17.4
TREES	7.9	0.0	11.6	72.7	7.8
FARM	7.8	0.0	48.0	36.1	8.1

Table 2.7: Contingency table for secondary image ML classifications

Est.true class	Class assigned by ML (% true class)				
	SEA	CLOUD	URBAN	TREES	FARM
SEA	34.5	2.1	62.8	0.0	0.6
CLOUD	n.a.	n.a.	n.a.	n.a.	n.a.
URBAN	0.0	52.5	46.2	0.1	1.2
TREES	0.0	0.6	41.9	42.8	14.7
FARM	0.0	58.4	41.2	0.1	0.3

To produce a "true" classmap, each of the image's 327,680 pixels would have needed to be ground truthed, which was impractical. An approximation to this was constructed by performing an ML classification using very carefully sub-divided spectral classes. The approximation is only valuable in this test because biases induced by this method of "true" classmap generation should favour the ML classification, but despite this bias, ML is still inferior to PROC in this example. The diagonal elements of the contingency matrices indicate that the PROC method is consistently better than ML at classifying an image using statistics generated in another image.

To investigate the behaviour of the algorithms, maps were produced of the discriminant values on which the classes were assigned (figures 2.13 – 2.17). For the PROC methods using weighted *D*-values, these maps are designed to show a visual representation of the unweighted *D*-values. Since the values span a very large range, a common approach would be to display log D, but since the 4.5th root of D was already calculated for the hybrid technique, this value is shown appropriately scaled in accordance with

$$D^* = (255 - 20.\sqrt[4.5]{D}) \tag{2.27}$$

with values restricted to lie in the range (0,255) by truncation. Similarly to display the discriminant values used by the ML scheme the values were transformed for display by

$$P^* = (255 - 20.\log P) \tag{2.28}$$

Figures 2.13 to 2.17 each comprises 5 images which are maps corresponding to a particular class. Under the transformations given by (2.27) and (2.28), a pixel is white if it has a high likelihood of coming from the class represented by the map.

Inspection of the figures reveals that for ML classification, water is classified with certainty, while for the simple PROC classification, the water discriminant function identifies itself correctly but not uniquely. The ML probability values for the secondary image indicate that classifications are doubtful, while PROC values have a similar range of magnitudes for the primary and secondary image classifications.

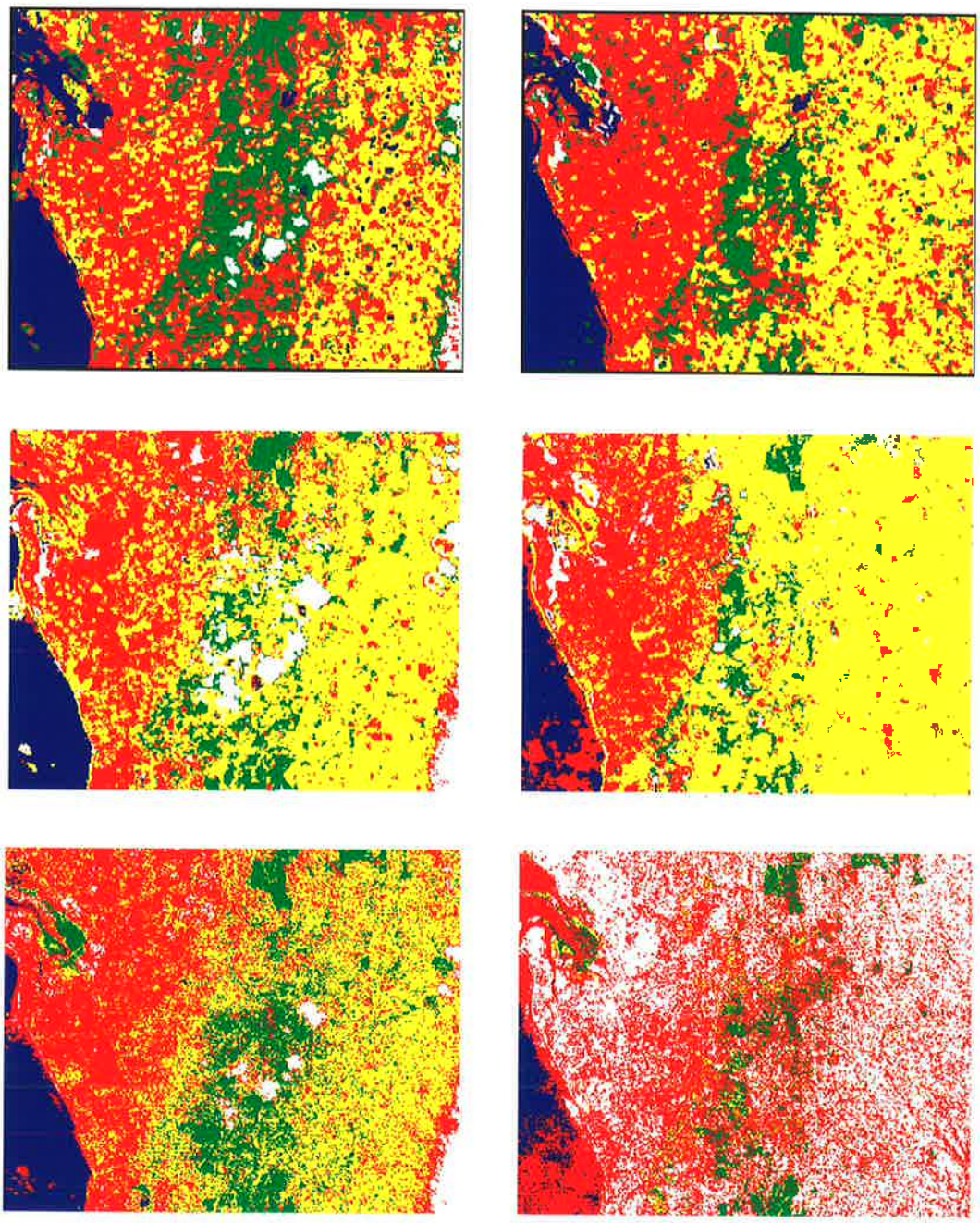


Figure 2.11: Classification of primary and secondary images using D-values (top row), hybrid classifier (second row), ML (bottom row). Colour indicates class: water=blue, cloud=white, urban=red, trees=green, farmland=yellow

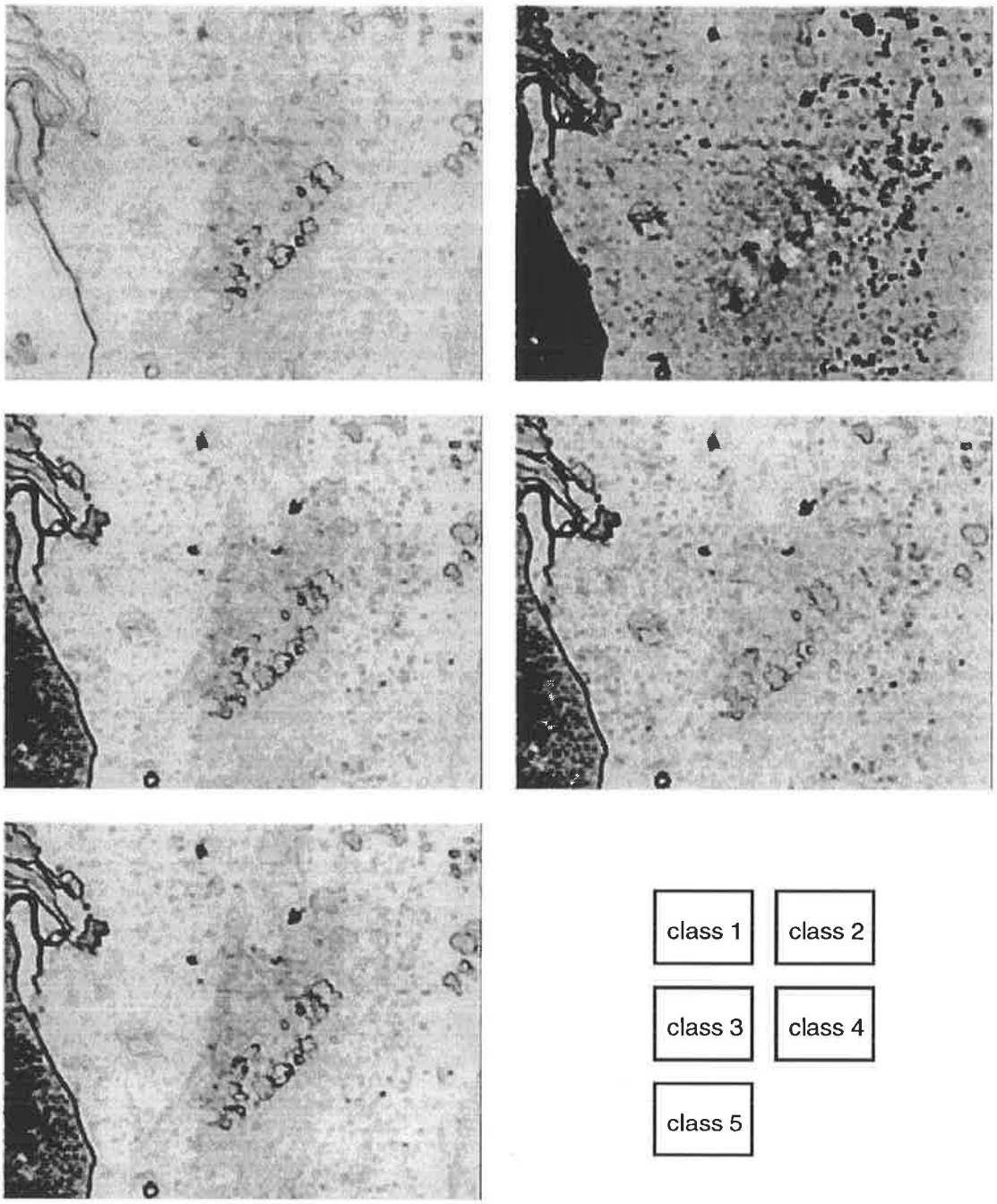


Figure 2.12: D^* value maps from primary image classification

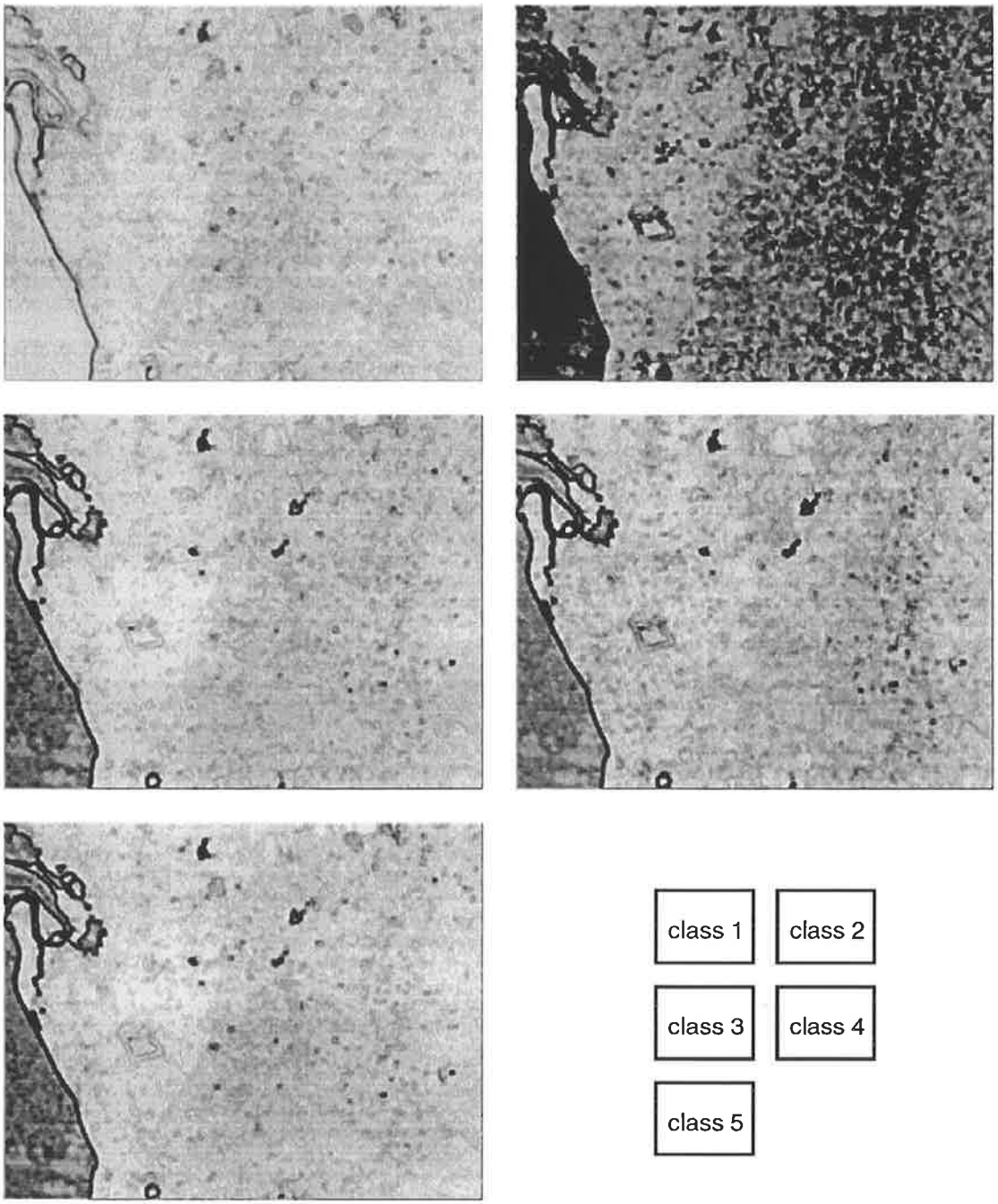


Figure 2.13: D^*-value maps from secondary image classification

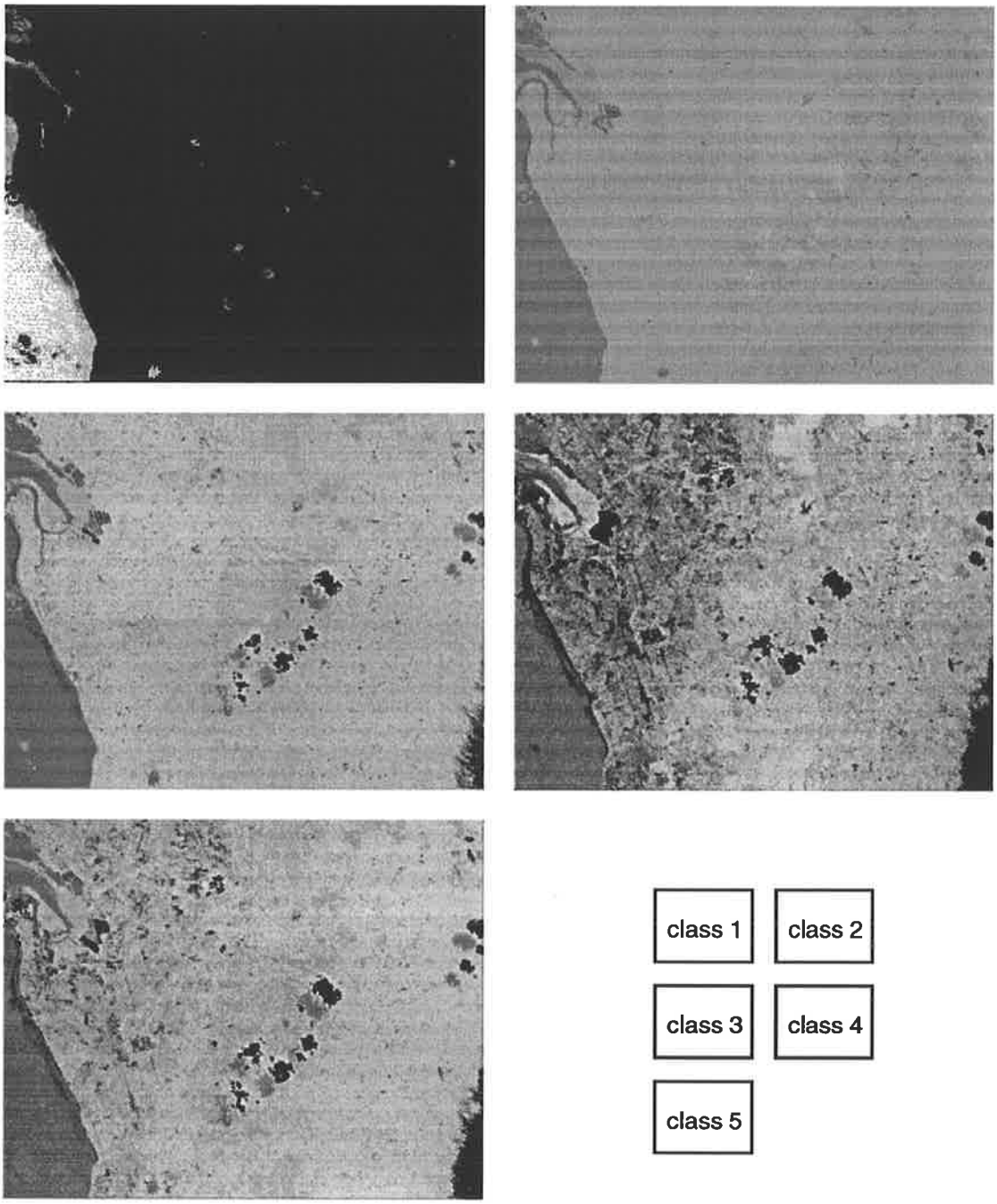


Figure 2.14: Probability maps from ML classification of primary image

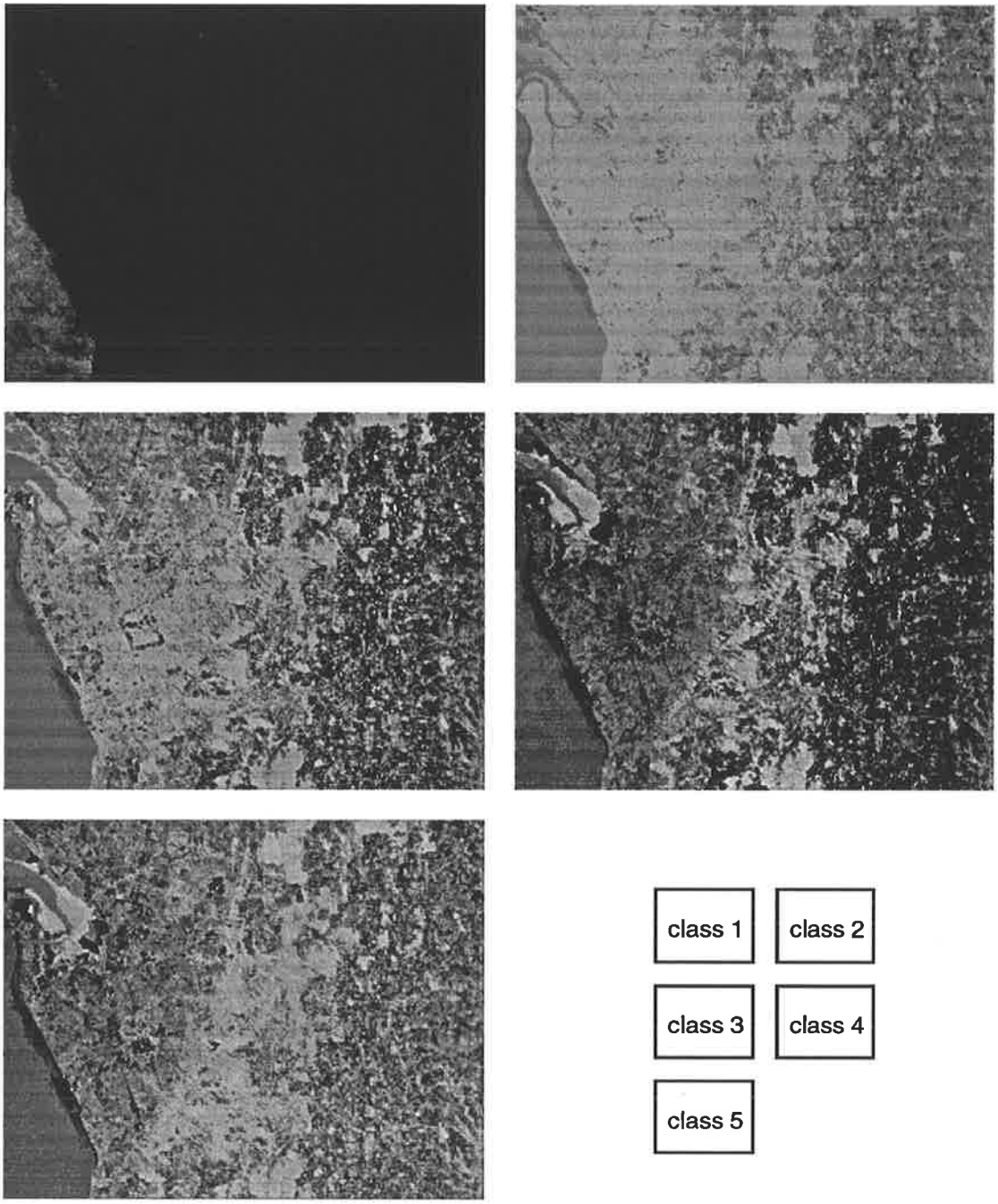


Figure 2.15: Probability maps from ML classification of secondary image

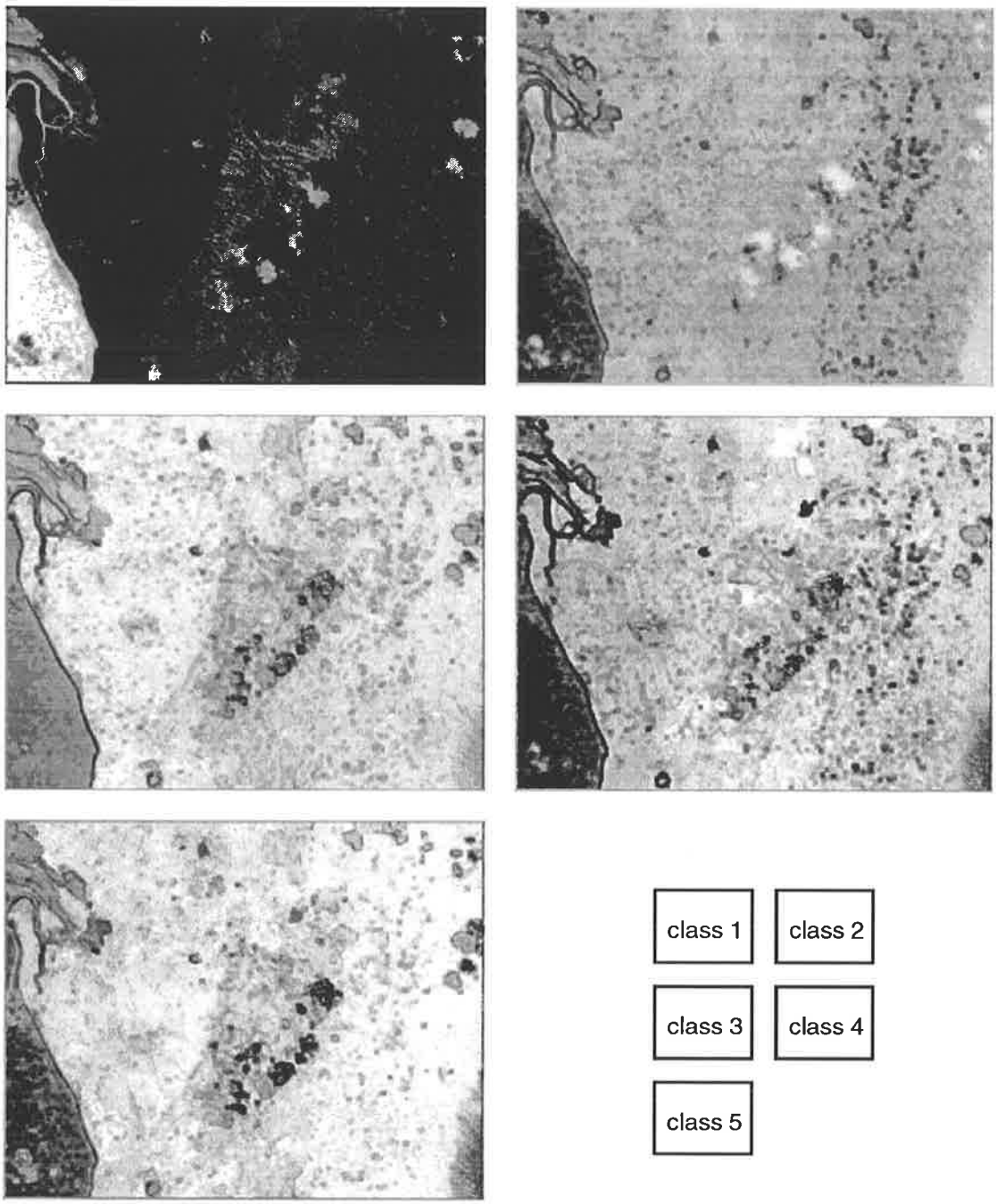


Figure 2.16: Probability maps from hybrid classification of primary image

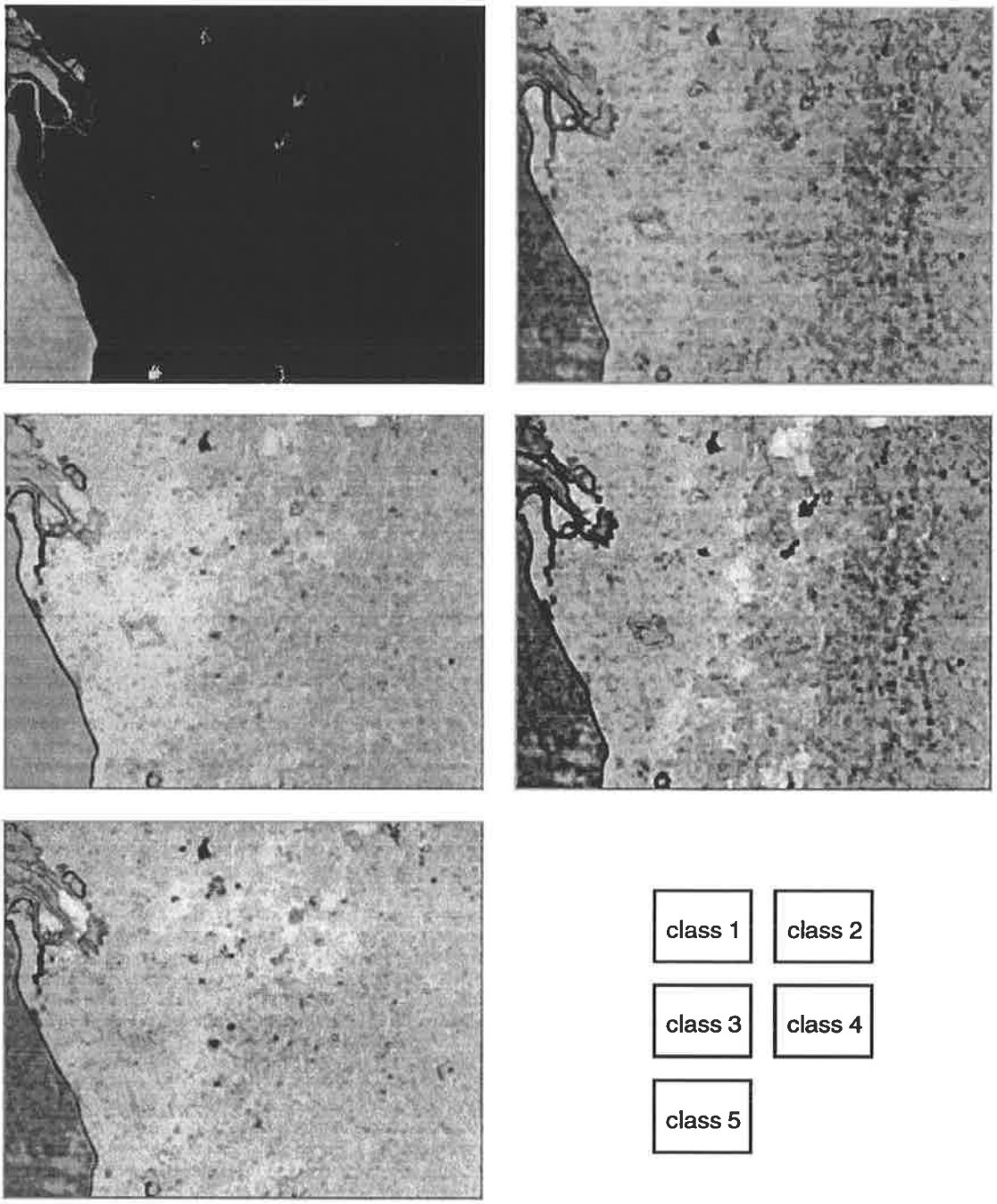


Figure 2.17: Probability maps from hybrid classification of secondary image

2.4 Discussion

2.4.1 The results.

In experiments using Landsat MSS data, the PROC method has shown itself capable of extracting information from an image not available using a conventional ML classifier. In particular, it can produce a sensible classification of an image using statistics generated from a different image.

It is self evident that because of the value of D is not affected by changes in $S_i I_{kl,i} r_{lc,i}$ (illumination and sensor gain) that multiplication of the values of radiance in a particular band by a constant should not change the resulting classification. The most important reason for needing to re-learn spectral signatures between scenes is to take account of the varying illumination. PROC is successful in re-using spectral signatures between scenes because of the insensitivity to changing illumination. However, while independence from multiplicative factors w.r.t. illumination also suggests that PROC should be able to classify area in shadow, where the illumination and spectral content are different but constant, this process appears to be sensitive to the level of additive noise in the image. Further, cloud shadow is likely to contain pixels which have values so small that they digitized to the zero level, suppressing any information they may carry. Because of these factors, the conclusion must be that this metric is not suitable for classifying areas in cloud shadow.

Why can weighted D-values be used to classify an image? D-values incorporate information about the similarity of the covariance matrix of a neighbourhood of a pixel to be classified to the neighbourhoods of pixels in the training set. This information describes the spectral variation within a compact spatial area. The need to weight D-values arises because in a practical classification system it is necessary to be able to classify pixels which are not in neighbourhoods solely populated by examples of the same class. Hence, the measured weighting factors incorporate some information about the diversity of the neighbourhoods in which one can expect to find pixels of a particular class.

D-values can be used usefully to augment radiance values as input to a ML classification scheme because they incorporate spatial information that is not available in normal ML classification schemes. Moreover, the D-values are not strongly correlated to the radiance data and hence provide more information. Transformation of D-values for the hybrid scheme does not affect the information content and is only for convenience of constructing a simple ML classifier.

In a simple experiment to incorporate spatial information, an ML scheme was implemented in which all radiance values from a neighbourhood of the pixel of interest were used in a stacked vector -- for a 3x3 neighbourhood this gives a 36-dimensional vector for a four band image. The strong correlation between neighbouring pixels' radiances is not unexpected, being part of the assumptions of any Markov field model. However the practical result of these correlations is that it is not possible to form a meaningful discriminant function because the covariance matrix has no true inverse.

2.4.2 Implementation issues - complexity, speed and parallelism

Calculation of D-values takes a long time compared to standard ML discriminant functions. The basic calculation of the ML discriminant function is

$$(\boldsymbol{x} - \overline{\boldsymbol{x}_r})^{\mathrm{T}} \boldsymbol{C}_r^{-1} (\boldsymbol{x} - \overline{\boldsymbol{x}_r}) \tag{2.29}$$

where $\overline{\boldsymbol{x}_r}$ and $\boldsymbol{C}_r^{-1}$ are pre-computed during training, giving a complexity of $O(M^2 + M)$ where M is the dimensionality of $\boldsymbol{x}$. For the calculation of D the basic calculation is

$$\sum_{i}^{M} \sum_{j}^{M} \left\{ \left(\boldsymbol{C}_r^{-1}\boldsymbol{C}_x - \boldsymbol{I}\right)_{ij} \left(\boldsymbol{C}_r^{-1}\boldsymbol{C}_x - \boldsymbol{I}\right)_{ji} \right\} \tag{2.30}$$

where $\boldsymbol{C}_r^{-1}$ is pre-computed during training giving a complexity of $O(M^3)$. It is also necessary to compute the covariance matrix for every pixel neighbourhood, which has a complexity $O(N^2)$ where N is the number of pixels in a neighbourhood. This may be lowered to $O(N)$ with a suitable intermediate memory store and processing sequence for a rasterized image.

Schemes of classification which compare values of discriminant functions can always be speeded up by simultaneous evaluation of the discriminant functions. Hence both ML and PROC could benefit from having one processor per discriminant function. Further speed up can be achieved by pipelining the calculation of the discriminant functions. Massively parallel architectures implementing artificial neural networks offer possibilities of speed up with relatively unsophisticated programming. These systems have other advantages associated with learning systems, especially learning of distributions.

2.5 Summary and conclusions

This chapter has described an implementation of a new classifier for image data based on pattern recognition by observation correlations (PROC). This classifier was based on the measurement of similarity of covariance matrices of collections data collected in neighbourhoods.

A simple formulation of PROC for classification of images was presented but found to require some tuning to give good results. New results were presented about the nature of the distribution the covariance similarity measure D and these results allowed the development of a hybrid algorithm combining data from a PROC processed image with raw spectral data to improve classification accuracy. By virtue of including data about a neighbourhood, the hybrid algorithm increases accuracy in examples tested

One of the original motivating problems for this application of the PROC algorithm, the transfer of spectral signatures from one image to another, has been shown to be valid but the resulting accuracies are not sufficient for a production system. The other motivating idea of classifying areas of the ground affected by cloud shadow was not shown to be viable. This was ascribed to quantization effects in the analogue to digital conversion of data; when low valued radiances fall below the lowest measurable level their information is lost.

The amount of computation required to extract covariance similarity measures from image data is an important problem when applying of this method. Implementation using some

parallel architectures was considered. The renewed interest in artificial neural networks as a paradigm for parallel computational machines leads to the subject of the next chapter. Artificial neural networks fulfil the requirement for a density-function-independent classifier, appear ideally suited to the task of generation of discriminant functions and have also been of interest in classification based on spatial characteristics.

Table 2.8: The first 8 moments (m1…m8) of D , for N = 100, estimated over 10000 synthetic samples

Bands	Mean	Variance	m3/m2	m4/m2	m5/m2	m6/m2	m7/m2	m8/m2
1	1.018	2.232	3.358	22.07	183.31	1811.	19790.	229881.
2	3.041	6.961	2.120	11.59	77.89	693.6	7247.	82973.
3	6.011	13.36	1.453	6.631	27.13	142.2	818.4	5144.
4	10.01	22.67	1.187	5.481	19.48	95.11	499.6	2883.
5	14.92	34.55	1.027	4.891	14.86	66.17	302.3	1556.
6	20.98	48.93	0.918	4.578	12.95	57.06	248.5	1228.
7	28.11	69.33	0.716	3.737	7.827	30.49	99.33	390.4
8	35.95	88.87	0.759	4.141	10.02	43.08	166.5	756.7
9	45.04	113.32	0.732	4.265	11.34	57.24	285.1	1710.
10	55.23	137.87	0.574	3.500	6.212	25.77	77.06	308.4
11	66.12	167.51	0.541	3.388	5.628	24.19	72.35	310.5

Table 2.9: The first 8 moments (m1…m8) of log(D), for N = 100, estimated over 10000 synthetic samples

Bands	Mean	Variance	m3/m2	m4/m2	m5/m2	m6/m2	m7/m2	m8/m2
1	-1.333	4.949	-1.399	6.126	-23.55	121.4	-710.8	4708.
2	0.713	0.928	-0.810	4.300	-10.62	45.26	-172.8	771.9
3	1.600	0.402	-0.506	3.685	-6.218	29.63	-90.43	411.1
4	2.182	0.234	-0.306	3.351	-3.341	21.08	-40.24	203.4
5	2.637	0.155	-0.225	3.178	-2.307	17.73	-24.33	142.4
6	2.995	0.113	-0.190	3.205	-2.110	18.78	-27.03	173.0
7	3.293	0.085	-0.109	3.111	-1.444	17.16	-20.88	150.9
8	3.549	0.068	-0.056	3.116	-0.670	16.82	-9.748	134.5
9	3.777	0.055	-0.075	3.165	-0.673	17.22	-5.792	132.6
10	3.985	0.045	-0.036	3.012	-0.303	14.69	-2.627	95.4
11	4.169	0.039	0.001	3.047	0.130	15.35	1.886	105.6

Table 2.10: The first 8 moments (m1...m8) of square root(D), for N = 100, estimated over 10000 synthetic samples

Bands	Mean	Variance	m3/m2	m4/m2	m5/m2	m6/m2	m7/m2	m8/m2
1	0.801	0.382	1.124	4.488	13.34	52.17	213.3	946.1
2	1.589	0.475	0.576	3.363	6.019	23.67	69.77	264.4
3	2.340	0.497	0.465	3.433	5.318	23.88	65.60	264.3
4	3.083	0.527	0.461	3.428	5.225	24.30	67.12	280.8
5	3.810	0.556	0.385	3.275	4.038	20.23	45.00	188.3
6	4.530	0.568	0.359	3.240	3.780	20.18	44.78	200.8
7	5.246	0.582	0.346	3.269	3.747	20.75	45.39	209.2
8	5.937	0.590	0.324	3.258	3.659	20.53	45.37	207.2
9	6.671	0.604	0.313	3.254	3.331	19.75	38.49	181.3
10	7.377	0.633	0.313	3.144	3.185	18.06	34.87	159.4
11	8.104	0.634	0.243	3.072	2.398	16.34	24.89	126.0

Table 2.11: The first 8 moments (m1...m8) of 4th root(D), for N = 100, estimated over 10000 synthetic samples

Bands	Mean	Variance	m3/m2	m4/m2	m5/m2	m6/m2	m7/m2	m8/m2
1	0.821	0.122	0.134	2.589	1.375	10.42	12.19	57.9
2	1.233	0.077	-0.010	2.987	0.385	15.66	9.70	129.2
3	1.514	0.056	0.024	3.125	0.566	16.69	8.35	124.6
4	1.737	0.044	0.046	3.147	0.304	17.69	0.58	149.9
5	1.940	0.036	0.078	3.084	0.805	16.03	7.96	116.0
6	2.118	0.031	0.079	3.058	0.705	15.38	6.25	104.1
7	2.281	0.028	0.087	3.130	1.032	16.98	13.25	131.5
8	2.436	0.025	0.134	3.088	1.376	16.18	13.87	117.9
9	2.580	0.022	0.091	3.038	0.943	15.28	9.48	105.0
10	2.711	0.021	0.088	3.006	0.694	15.10	6.04	106.4
11	2.843	0.019	0.100	2.971	0.967	14.70	9.29	99.2

Table 2.12: The first 8 moments (m1...m8) of 4.5th root(D), for N = 100, estimated over 10000 synthetic samples

Bands	Mean	Variance	m3/m2	m4/m2	m5/m2	m6/m2	m7/m2	m8/m2
1	0.833	0.103	-0.030	2.567	0.196	9.746	3.746	47.8
2	1.201	0.057	-0.148	3.027	-1.221	15.31	-10.91	106.0
3	1.442	0.039	-0.002	3.042	0.109	15.40	2.336	107.9
4	1.639	0.030	0.002	3.119	-0.152	16.67	-3.018	127.0
5	1.805	0.025	0.084	3.122	0.823	16.31	7.575	116.0
6	1.949	0.022	0.119	3.113	1.266	17.10	15.62	143.5
7	2.083	0.018	0.069	3.150	0.952	17.43	12.62	139.5
8	2.207	0.016	0.144	3.124	1.538	17.07	16.62	133.6
9	2.321	0.014	0.141	3.013	1.442	15.86	15.71	126.2
10	2.428	0.014	0.112	3.033	1.122	15.51	11.81	110.9
11	2.530	0.012	0.118	3.059	1.243	15.89	13.98	117.2

Table 2.13: Expected values of 5th root of D and its first 8 moments (m1...m8), for N = 100, (estimated over 10000 synthetic samples)

Bands	Mean	Variance	m3/m2	m4/m2	m5/m2	m6/m2	m7/m2	m8/m2
1	0.841	0.088	-0.105	2.640	-0.341	10.46	0.348	53.8
2	1.175	0.045	-0.178	3.169	-1.616	16.69	-14.31	116.6
3	1.393	0.031	-0.134	3.105	-1.079	16.01	-9.172	111.7
4	1.558	0.022	0.010	3.124	0.240	16.79	2.802	128.6
5	1.698	0.018	0.013	3.063	0.069	15.47	-0.589	105.7
6	1.824	0.015	0.062	3.089	0.662	16.33	7.604	122.2
7	1.934	0.013	0.100	3.000	0.789	14.72	6.622	98.2
8	2.036	0.011	0.100	3.098	1.047	16.55	12.14	127.1
9	2.134	0.010	0.079	3.025	0.790	15.46	8.897	114.4
10	2.221	0.009	0.078	3.082	0.776	15.83	8.103	110.9
11	2.308	0.008	0.136	3.032	1.472	15.82	16.76	119.3

Table 2.14: Theoretical values of functions of D's mean for $N = 100$

Bands	$E\{D\}$	$E\{\log(D)\}$	$E\{\sqrt{D}\}$	$E\{\sqrt[3]{D}\}$	$E\{\sqrt[4]{D}\}$	$E\{\sqrt[5]{D}\}$
1	1.010	0.010	1.005	1.003	1.002	1.002
2	3.030	1.109	1.741	1.319	1.279	1.248
3	6.061	1.802	2.462	1.569	1.492	1.434
4	10.101	2.313	3.178	1.783	1.672	1.588
5	15.152	2.718	3.892	1.973	1.829	1.722
6	21.212	3.055	4.606	2.146	1.971	1.842
7	28.283	3.342	5.318	2.306	2.102	1.951
8	36.364	3.594	6.030	2.456	2.222	2.052
9	45.455	3.817	6.742	2.597	2.335	2.145
10	55.556	4.017	7.454	2.730	2.442	2.233
11	66.667	4.200	8.165	2.857	2.543	2.316

Chapter 3

ANNs for Supervised Classification of Multi-Spectral Images

Summary: *In this chapter, architectures for the classification of multi-spectral images are examined and the multi-layer perceptron chosen as the most appropriate. With suitable training, a neural network based on a three layer perceptron is theoretically capable of classifying patterns from sets with arbitrarily distributed density functions in N-dimensional space. A multi-layer perceptron, trained using back propagation, is shown to be capable of classifying pixels represented by patterns in 4-dimensional colour space with an accuracy comparable with the ML scheme. Some domain-specific implementation guide-lines are developed. Results from neural network classification are presented and compared with classification made using the more conventional maximum likelihood technique. Finally, the internal representation in a trained multi-layer perceptron is analyzed to find out what it learnt about the training sets.*

3.1 A flexible model-independent classifier

3.1.1 Overview

Since the recognition of colour texture does not seem to be a difficult problem for the human brain it seems obvious to pose it as a problem for a simulated human brain – an artificial neural network. This chapter describes the first stage – the multi-spectral part – in developing an ANN for spectral/spatial classification. We concentrate on the classification process where pattern vectors to be processed come from individual pixels, but admit cases where some components of the vectors may have been derived by preprocessing.

Revived interest in artificial neural networks, due at least in part to the work of Hopfield and Tank [Hopfield86] and Rumelhart *et al.* [Rumelhart86], has inspired a number of researchers to examine the possibility of creating new image-recognizing architectures. An artificial neural network (ANN) is a biologically inspired paradigm for computing, in which processing is done by a network of interconnected co-operating nodes. In most ANNs, nodes have very limited

arithmetic power, and the information in the network is concentrated in the values of interconnections between nodes.

Amongst many aspects which make ANNs appealing for image processing are the inherent parallelism, the biological plausibility coupled with the fact that most humans are better at vision than machines, and the adaptive nature of networks which allows them to learn from example. The non-parametric nature of ANN learning suggests that they could also provide a useful approach to the design of a general classifier for remotely sensed data containing features with unknown distributions. The microscopic level of the parallel structure of ANNs also makes possible adding more parallel processors to handle extra input dimensionality without any speed penalty. However, the adaptation and learning of ANNs is not achieved without cost; in some cases the increased time taken for a network to learn a distribution will represent a significant and unacceptable cost. While the attitude taken in the work described here is that affordable computer power is increasing[1], this chapter includes a sober analysis of the cost in terms of time and resources of using the ANN approach

This chapter is structured as follows. The remainder of 3.1 introduces a variety of ANNs for classification, gives reasons for the selection of a particular type, the multi-layer perceptron (MLP), and reviews the work by others on the use of MLPs, much of which was conducted in parallel with the work described in this thesis. Section 3.2 defines an MLP, shows how it can be trained for classification tasks and discusses its classifying capabilities. Section 3.3 discusses the implementation of a trainable MLP, and presents a thorough exploration of kind of structure suitable for classification of multi-spectral data. Results of spectral classifications are presented and compared to ML classifications. Section 3.4 presents an investigation into the internal representation developed by a successfully trained MLP. Finally section 3.5 concludes with a discussion of the relative merits of the MLP as a classifier compared to ML.

From the experimental work presented in this chapter, it will be seen that MLPs are capable of performing at least as well as conventional classification (ML) in all situations, and are capable of out-performing ML, given appropriate tuning parameters. An MLP can only achieve better results than the ML when the underlying model deviates from the multivariate normal, which evidently, it does in most cases.

Of course, for the practical use of ML, prior to measuring class parameters it is normal to seek to expose multimodality of a class distribution using clustering and manual intervention. Under these circumstances ML works well. However, when using an MLP, there is no need to examine the data in this way before extracting the class characteristics and manual intervention is not required.

---

[1]During the time taken to carry out the work described in this thesis, the typical time to train an ANN to recognise spectral signatures from a 4-band MSS changed from 8.5 hours on a Vax 11/780 to 15 minutes on a Sun Sparc2.

3.1.2 Specification of a suitable ANN

The ideal multi-spectral classification system for remotely-sensed data is one that learns class attributes from examples and is able to use the attributes to classify previously unseen images. A block diagram of an idealised scheme is shown in figure 3.1. For each class ω_r a set of reference pixels is used to train the network to have the rth output = 1 and all other outputs = 0. The elements of a pixel's pattern vector form the inputs. The network develops its own internal representation during training by adjusting a set of parameters θ. During classification a feature vector presented to the ANN is classified according to the output with the maximum value.

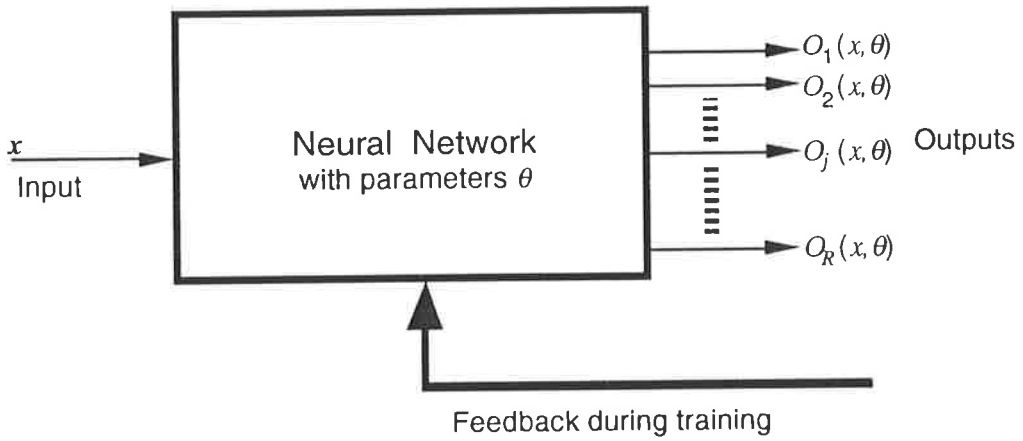


Figure 3.1: An ANN for classifying multi-spectral data.

The functions we would expect from an ANN providing this role would be *feature selection*, *class characterization* and finally *classification*. These functions are not necessarily distinct even in classical pattern recognition systems [Young74] and may be represented internally in an ANN in a way that is inseparable on inspection from outside.

In formulating the task, there is an important assumption that input data from the same class will be clustered in feature space, and hence it will be possible to partition feature space into regions which may be linked to classes. Only weak assumptions about the shape of the clusters are considered acceptable and there is no restriction on the number of clusters that may map to one class although conversely a cluster may only belong to one class. Figure 3.2 shows data from reference areas identified in a Landsat MSS image. The data is plotted as a 3 dimensional scattergram with the axes corresponding to the radiances of three least correlated spectral bands. The points are colour coded to indicate from which reference set they are taken. Clearly there is a tendency for data from the same class to form clusters in colour space. A more precise illustration of this may be seen in figure 3.3 which shows a scatter plot matrix diagram of the data

Summarising, the specification is that the network should be able to take as input, a multi-dimensional vector containing components with continuous (or pseudo-continuous) values, provide an output of a chosen class, and be trainable by example, i.e. learn within a supervised learning scheme.

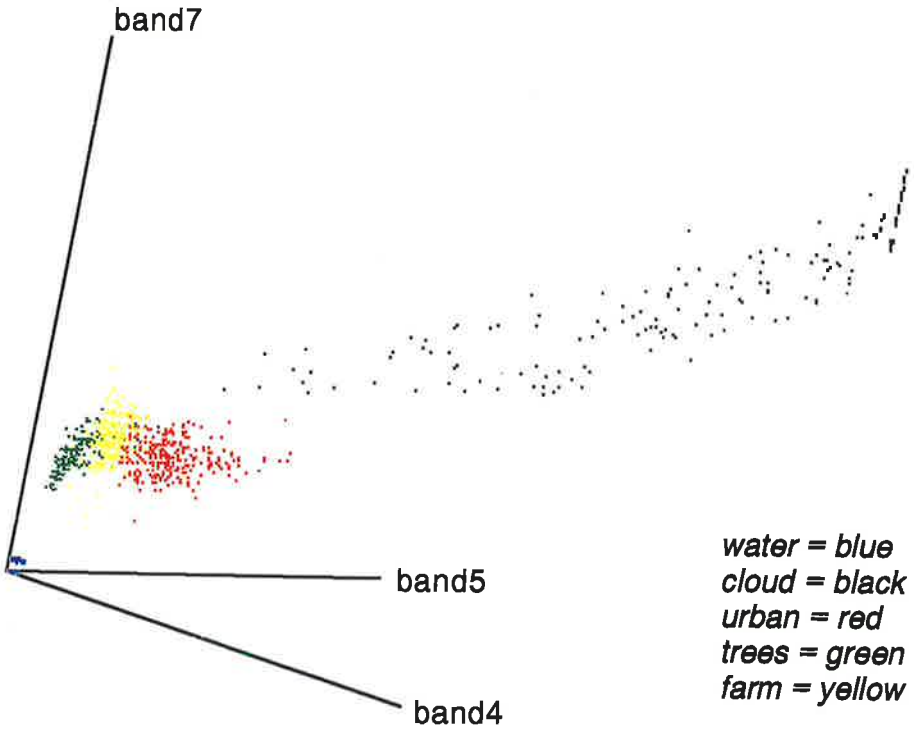


Figure 3.2: 3-D scatter-plot of data from Landsat MSS reference areas

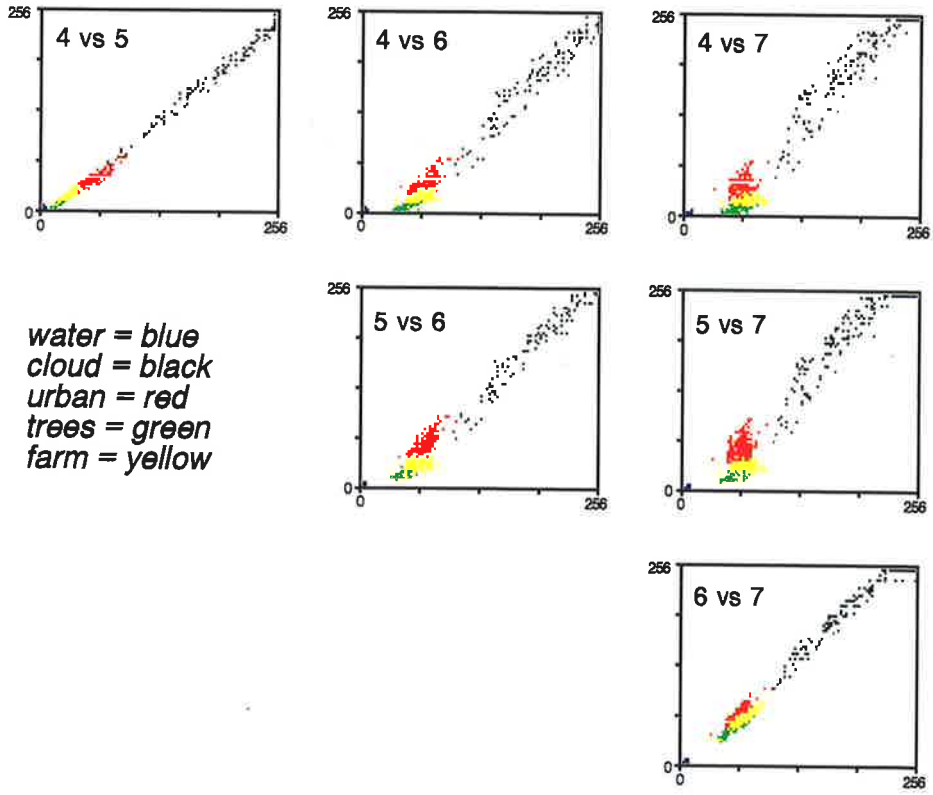


Figure 3.3: 2D Scatter plots of data from Landsat MSS reference areas
A point is plotted at position (band*x*, band*y*) for each reference pattern

3.1.3 Selection of a suitable ANN

Many of the commonly known ANN architectures are unsuitable for spectral classification because they are designed for binary inputs only. Examples include the Hopfield Net [Hopfield86] and the Adaptive Resonance Theory 1 network (ART1) [Grossberg87a]. The Boltzman Machine [Hinton86] was also deemed unsuitable because of the discrete nature of its input states. Effectively one input state is needed to represent each possible input vector. Boltzman machines are being studied for use in monochrome image restoration (e.g. [Longstaff90]) where the number of possible grey levels is restricted, but the number of possible vectors in a multi-band image is potentially very large.

Other architectures were considered unsuitable because they use unsupervised learning. These included ART2 [Carpenter87], the Self Organizing Map [Kohonen87], the Neocognitron [Fukushima86] and the Binary Associative Network [Kosko88]. However, the use of unsupervised ANNs for image processing is not uncommon: Grossberg, Kohonen, Fukushima and others have applied such ANNs to certain computer vision tasks. Computer vision is related to the classification of remotely sensed images, but the purpose of the processing described here is not to emulate human vision. Remotely sensed images can include more information than can be used by a human. The most obvious example is increased spectral bandwidth, but more subtle advantages exist. For example, there is the potential to make better use of colour information with high spatial resolution since it is not subject to recoding and consequent spatial frequency bandwidth limiting that occurs in human neural processing of visual information [Granrath81]. The use of unsupervised networks may be of use for clustering; i.e. for pre-processing data to discover "natural" groupings.

The artificial neural network variously known as the Gamba-perceptron, Multi-layer Perceptron (MLP) and very loosely as the Back Propagation Network (BPN), accepts as input continuous-valued data, is able to learn complex distributions under supervision and is able to indicate a classification at its output. It is well matched to the classification of remotely sensed data. The MLP was examined in the 1960's [Nilsson65][1] and early 1970's [Minsky69] (with handwritten alterations in the 1972 printing) but only became useful when Rumelhart *et al.* [Rumelhart86] published a semi-empirical method for training. Many of the results in this thesis rely on an MLP trained using Rumelhart's method.

Despite the apparent mismatch between other ANN architectures and the multi-spectral data classification task, a number of researchers have attempted to use a diverse cross-section ANNs by reformulating the problem. Eklundh *et al.* proposed a system of using associative nets which attempted to classify regions by combining spectral and contextual information [Eklundh86]. The input values were recoded by requantization to 32 levels and separate input node was used for each level on each input. It was claimed that the system, which used unsupervised learning, needed no statistical model, with a clustering procedure inherent in the

[1] Nilssen formulated a ML classification scheme in the same terms as a neural network, but emphasising simplicity of structure rather than uniform simple function at the node. His work required the effective use of higher order node which is not addressed in this thesis.

learning stage. The use of unsupervised learning relies on clustering criteria which often embody assumptions about the form of the input data in a hidden way. A particular clustering algorithm may impose an unrecognized distorted topology on the data causing unnatural groupings to occur. In Eklundh's paper (and many other ANN papers) it is necessary to rely on the biological plausibility of the model used to perform clustering to ensure that groupings of data are sensible[1].

Even in using the MLP, some researchers have found implementation easier by reformulating the problem. Benediktsson *et al.* used an MLP for sensor fusion, combining multi-spectral data and terrain elevation data [Benediktsson89] but reformulated the problem as one of classifying *binary* data into one of R classes. In their 1989 paper, they specified that 8-bit binary coding of the input data channels had been used as direct input to an ANN with 8 bits of input for each band or data source. Benediktsson's work was performed in parallel with the work described in this thesis, and publication of the 1989 results led to the investigation of input representation; it appears that it should be harder to generalize examples when using such an arbitrary input space, even given that they used "gray" coding for their input representation [Benediktsson90]. Benediktsson *et al.* have since developed a new network architecture [Benediktsson91] specifically for the classification of multi-source data.

A number of other researchers have also carried on parallel work in this area. Sheldon describes the Satellite Image Analysis using Neural Networks system (SIANN) in use by NASA to automatically classify images by attributes suitable for rapid retrieval from databases [Sheldon90]. Kanellopoulos *et al.* report success in using an MLP to classify SPOT multi-spectral data formed by combination of two registered images [Kanellopoulos91]. Hepner *et al* discuss minimizing the extent of the required training sets in a preliminary report on their work [Hepner90]. Questions that they posed as further research about an MLPs internal representation and the ability to handle spatial information have been covered by the work reported in this thesis.

Blanz and Gish [Blanz90] [Gish89] [Gish90] have investigated the use of ANNs for monochrome image segmentation, using multi-dimensional data based on derived image features. This is discussed further in Chapter 4.

3.2 Multi-Layer Perceptrons for classification

3.2.1 Structure

A multi-layer perceptron (MLP) is here considered to be a feed-forward network constructed entirely of interconnected nodes that perform a non-linear function of the weighted sum of their inputs. A single node's output, x', is defined by

$$x' = f(w^{\mathrm{T}}\boldsymbol{x}) \tag{3.1}$$

[1]The models used are of necessity simplified and there can be no guarantee that in such a non-linear system that simplifications will not markedly affect behaviour.

where f is a non-linear function, $\boldsymbol{x}$ is the augmented state vector which includes a constant component (1) plus all inputs x_i to the node

$$\boldsymbol{x} = (1, x_1, \ldots x_i, \ldots x_N)^T$$

and $\boldsymbol{w}$ is the weights vector for this node,

$$\boldsymbol{w} = (w_0, w_1, \ldots w_i, \ldots w_N)^T$$

where w_i is the weight applied to the i th input and w_0 acts as a threshold for this node. A typical node is depicted in figure 3.4.

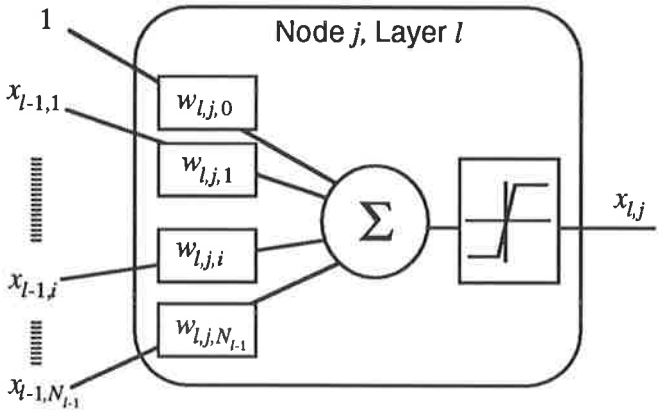


Figure 3.4: A perceptron node.

The non-linear function f is most often determined by the requirement to make the network trainable. This chapter considers networks which can be trained using the modified gradient descent method known as back propagation [Rumelhart86], which use the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

The basic network structure of an MLP comprises a feed forward network or directed graph (figure 3.5) with each node performing the function described in equation (3.1). For L layers, and J_l nodes in each layer, the output of the j th node in layer l becomes

$$x_{lj} = f\left(\boldsymbol{w}_{lj}^T . \boldsymbol{x}_{l-1}\right) \tag{3.3}$$

In this formulation $\boldsymbol{x}_0$ is the augmented vector of inputs.

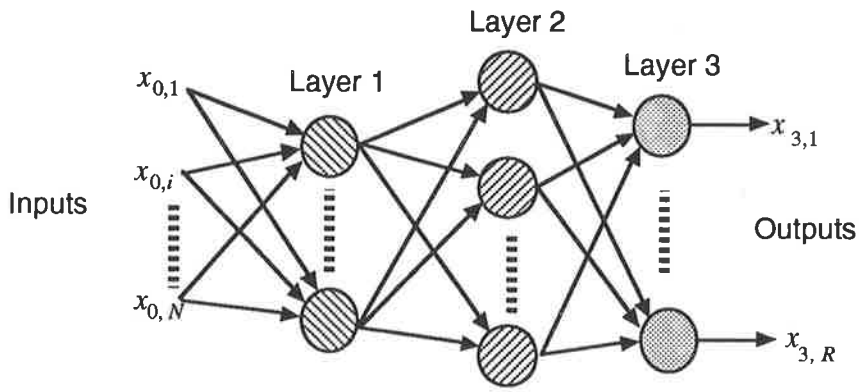


Figure 3.5: A typical perceptron-based neural network.

With ANNs of structures other than MLPs it has become convention to refer to the inputs as input nodes, which form a layer. A number of researchers (e.g. [Hecht-Nielsen90]) now refer to the above L-layer MLP as an L+1 layer network; for example, they would refer to figure 3.5 as a *four* layer network, despite the fact that there are only *three* active layers. Where confusion is possible, this thesis refers to "L (active) layers"[1]. Another way of describing the number of layers is to count the "hidden" layers. A hidden layer contains (active) nodes whose outputs are not accessible from outside the MLP. Where L=3, the network is referred to as having two hidden layers.

In an interesting variation of the MLP structure, inputs are connected to nodes in the first layer and, in addition, to nodes in the output layer. Connections of this kind are of interest, especially when used in an image classification application, because it is hypothesized that these connections allow the network to approximate simple groupings of class data in n-dimensional pattern space with fewer weight modifications. This configuration has been investigated theoretically by Sontag [Sontag91] and tried practically with multi-spectral data by He *et al.* [He90]. Neither theoretical nor practical results were able to establish definite advantage in this structure, but it is likely that the many unexplored interacting parameters may be masking potential gains.

3.2.2 Classification

In order to discuss training and MLP capabilities, it is necessary to define what "classification" means in the ANN context. Here it is defined in the broadest possible sense as the process which gives an output when a pattern is presented to input of the network. The general form of the output is a multi-component continuous-valued vector. The output of an MLP may be interpreted as a probability function [Bridle89], a competition score or simply as a yes/no vote, i.e an approximation to a boolean function (e.g. [Rumelhart86]).

In this most general sense an MLP performs a mapping from an input space of real valued vectors to an output space of real valued vectors

$$\mathcal{F}: \mathbb{R}^M \rightarrow \mathbb{R}^R$$

where there are M components in the input vector and R components in the output vector. In the MSS classification task the mapping has restricted input and output spaces, the specifics of which are stated in section 3.2.5.

3.2.3 Training

Experiments described in this thesis use back propagation [Rumelhart86] for adapting weights. Much of the literature tends to fuse (and hence confuse) the *multi-layer perceptron* structure, *generalized gradient descent* and *error back propagation*, to give "back propagation networks" (e.g. [Hecht-Nielsen90]). Here, these terms are used to describe different things; the

---

[1] This is the convention consistent with the work of Minsky and Papert [Minsky69], Widrow *et al.* [Widrow88]. This nomenclature is intuitively more appealing since cascading two 3 (active) layer networks gives a 6 (active) layer network, whereas in the other notation cascading two 4 layer networks gives a seven layer network.

MLP has been defined as a special case of feedforward network, back propagation (BP) will be defined as a method for determining the error signal which provides information on how each weight (especially those on hidden nodes) contributes to the output error, and generalized gradient descent will be used to describe the way in which the error signal is used to adjust the weights.

For an L layer network, training proceeds by applying an augmented input vector $\boldsymbol{x}_0$ to the input of the network, and comparing the resulting output vector $\boldsymbol{x}_L$ with the required output vector $\boldsymbol{d}$. Weights are adjusted to minimize a cost function $E(W)$ (similar to an energy measure) relating the errors at the output of the MLP to the values of the internal parameters W. A typical, but not exclusive, definition of E is the L^2 norm of the output error summed over all training patterns:

$$E(W) = \sum_{\mathbf{p} \in \mathbf{P}} \{E_{\mathbf{p}}(W)\} = \sum_{\mathbf{p} \in \mathbf{P}} \left\{ \sum_{i=1}^{M} (x_{L,i}^{\mathbf{p}} - d_i^{\mathbf{p}})^2 \right\} \tag{3.4}$$

where $x_{L,i}^{\mathbf{p}}$ is the ith output of the MLP when presented with training pattern $\mathbf{p}$, $d_i^{\mathbf{p}}$ is the corresponding desired output, and $\mathbf{P}$ is the set of all training patterns. Burrascano [Burrascano91] discusses a variety of candidates for $E()$, emphasising that the choice makes important assumptions about the expected distributions of the errors. For the case of the L^2 norm Burrascano showed that the assumption is of Gaussian distributed error, and is probably the best choice, given that the total error comprises sum contributions from many parameters and so is likely to be governed by the central limit theorem.[1]

Rumelhart *et al.* [Rumelhart86], [Rumelhart86a] showed that by using a recursive procedure (BP), then the "simple delta rule", a weight update rule due to Widrow and Hoff [Widrow60], could be extended to a "generalized" delta rule which gives adjustments for weights in hidden layers. The simple delta rule is given by

$$\Delta w_{ji} = \beta \sum_{\mathbf{p} \in \mathbf{P}} (d_j^{\mathbf{p}} - x_{Lj}^{\mathbf{p}})\, x_{0i}^{\mathbf{p}} \tag{3.5}$$

where Δw_{ji} is the change made to the weight on the connection from the ith node in layer L-1 to the jth node in layer L, $x_{0i}^{\mathbf{p}}$ is the ith element of the input pattern $\mathbf{p}$, and β is a carefully chosen gain factor, not necessarily constant during training. This rule requires that the error contributions from all patterns be measured before a weight update is made and was devised for networks with no hidden layers.

Rumelhart *et al.* noted that, while the Widrow-Hoff rule requires that weight adjustments are made with respect to $E(W)$, provided β is small, gradient descent can be approximated closely by making weight adjustments with respect to $E_{\mathbf{p}}(W)$. This allows a weight adjustment after presentation of each pattern ([Rumelhart86a], page 324) and simplifies the requirement for local memory at each node. Hence, the simple delta rule becomes

$$\Delta_{\mathbf{p}} w_{ji} = \eta\, (d_j^{\mathbf{p}} - x_{Lj}^{\mathbf{p}})\, x_{0i}^{\mathbf{p}} = \eta\, \delta_{Lj}^{\mathbf{p}}\, x_{0i}^{\mathbf{p}} \tag{3.6}$$

---

[1] Even though this is in part a non-linear sum, there is no obviously better choice of cost function.

where $\Delta_{\mathbf{p}} w_{ji}$ is the change made to the weight from the ith to the jth node after presentation of the training pattern **p**, η is a small constant gain factor, $\delta_{Lj}^{\mathbf{p}} = d_j^{\mathbf{p}} - x_L^{\mathbf{p}}$ is called the error signal and other symbols are as above. There is considerable discussion about the validity of this approximation amongst researchers, and it is further discussed in section 3.3 as an implementation issue.

Rumelhart *et al.* observed that the simple delta rule was equivalent to minimisation of $E(W)$ by gradient descent in weight space for a one layer network with linear activation functions. For a multi-layer net with non-linear activation functions it is desirable to similarly perform, or at least approximate, gradient descent in weight space with an equally uncomplicated weight update rule.

Consider the intermediate value y (the input to the activation function) formed by the weighted sum of inputs within an MLP node. The forward transfer function of a node in layer l can be re-written (dropping the "**p**" superscript for convenience)

$$y_{li} = \sum_{j=0}^{N_{l-1}} w_{lij} f(y_{l-1j}) \tag{3.7}$$

where f is the activation function and w_{lij} are the weights from node j in the (l-1)th layer to node i in the l th layer, and N_{l-1} is the number of nodes in layer (l-1). The error function at the output layer can be written

$$E(w) = \sum_{i=0}^{N_L} (f(y_{Li}) - d_i)^2 \tag{3.8}$$

Suppose the values y_{ij} , for all i, are inputs to a truncated network starting at the lth layer. Then the sensitivity of the network to the input y_{ij} is given by $\frac{\partial E}{\partial y_{li}}$. Moreover, if we know $\frac{\partial E}{\partial y_{l+1i}}$ for all i, we have

$$\frac{\partial E}{\partial y_{li}} = \sum_{j=0}^{N_l} \frac{\partial E}{\partial y_{l+1i}} \frac{\partial y_{l+1i}}{\partial y_{li}}$$

$$\frac{\partial E}{\partial y_{li}} = \sum_{j=0}^{N_l} \frac{\partial E}{\partial y_{l+1i}} w_{l+1ji} f'(y_{li})$$

$$\frac{\partial E}{\partial y_{li}} = f'(y_{li}) \sum_{j=0}^{N_l} \frac{\partial E}{\partial y_{l+1i}} w_{l+1ji} \tag{3.9}$$

Define

$$\delta_{li} = \frac{\partial E}{\partial y_{li}}$$

$$\delta_{li} = f'(y_{li}) \sum_{j=0}^{N_l} \delta_{l+1j} w_{l+1ji} \tag{3.10}$$

Since w_{l+1ij} directly influences input y_{ij} into the truncated network, we have

$$\frac{\partial E}{\partial w_{l+1ji}} = \frac{\partial E}{\partial y_{li}} \frac{\partial y_{li}}{\partial w_{l+1ji}}$$

$$= \delta_{li}\ f(y_{li})$$

$$= \delta_{li}\ x_{li} \tag{3.11}$$

For output layer nodes, calculation of the error signal follows by taking

$$\frac{\partial E}{\partial x_{Lj}} = -\ (d_j - x_{Lj})$$

hence

$$\delta_{Lj} = (\ d_j - f(y_{Lj})\ )\ f'(y_{Lj}) \tag{3.12}$$

For a network in which each node has a sigmoid activation function, calculating the derivatives gives the error signal for output layer node j:

$$\delta_{Lj} = (\ d_j - x_{Lj}\ )\, x_{Lj}(1 - x_{Lj}) \tag{3.13}$$

and the error signal for a node j in an arbitrary layer l is given by the recursive calculation (error back propagation):

$$\delta_{lj} = x_{lj}\,(1 - x_{lj}) \sum_{i=1}^{N_{L+1}} \delta_{l+1\,i}\, w_{l+1\,ij} \tag{3.14}$$

While it is common to use a sigmoidal activation function, any monotonic increasing, continuously differentiable function would be suitable.

An untrained network has all weights set to randomized values. To learn from the presentation of a known pattern we adjust the weights into node i of layer l according to the generalized delta rule:

$$\Delta w_{lij} = \eta\, \delta_{li}\, x_{l-1\,j} \tag{3.15}$$

where η is the "learning rate". The value of learning rate must be chosen empirically as a compromise between speed of learning and probability of instability. Because we are going to allow a weight update after each pattern presentation, it is desirable to add a smoothing term to prevent single points having undue influence on the weights, giving:

$$\Delta w_{lij}(n) = \eta\, \delta_{li}\, x_{l-1\,j} + \alpha\, \Delta w_{lij}(n-1) \tag{3.16}$$

where α is a term known as "momentum" and n is a counter, incremented after each weight adjustment. Again it is necessary to select α and η empirically, and they are generally held constant during training. The values of α and η interact in way that is described phenomenologically by Tollenaere, who specifies a regime for altering α during training to speed the reduction of error [Tollenaere90]. The claim that it is possible to use higher learning rates with the modified update rule is consistent with the increased stability afforded by the momentum term, but this term will not obviously benefit learning from well-behaved training sets.

Finding the parameters of a system subject to a given mapping from input to output is a standard optimization problem with a number of highly developed methods available for solution (e.g. [Press86]). Learning by back propagation with gradient descent is attractive because of the local nature of calculations required which is consistent with the architectural goals of ANNs. The principle of error back propagation is not restricted to use with the steepest descent algorithm. Some researchers have used BP with minimisation techniques

from classical numerical methods. Conjugate gradient [Moller90], has been particularly favoured for the updating of weights during training.

Conventional techniques use knowledge about the distribution of the error, and typically use higher order derivatives of the mapping function to make good guesses for Δw. Where the function specification is given by a large set of data the need for continual re-evaluation of numerical derivatives can be very computationally intensive, making their application impractical. Inevitably there is a price for the speed up in terms of imposition of restrictions on the input space, or increasing risk of finding local minima in weight space. Such speed-ups are necessarily domain specific. At the current stage of development there is enough uncertainly about the gains of using speed-up techniques to make their inclusion in a classifying system *ad hoc*.

3.2.4 Capabilities of MLPs

The assessment of the classification ability of the MLP architecture is based on the degree of generality with which it can associate regions in feature space with classes. If we consider an MLP in which the activation functions are Heavisides[1] (step or threshold functions), then the interpretation quoted by Lippmann [Lippmann87] in his widely read review paper is useful. The first active layer in an MLP can form hyperplanes in pattern space, which can be formed into closed subspaces by the action of the second (active) layer. The third (active) layer can then perform the action of combining disjoint regions in pattern space to from the desired class. The partitioning of input space is represented in figure 3.6.

This explanation is not correct for an MLP using sigmoidal activation functions, since it takes no account of the extra approximating power available as a consequence of the unsaturated region of the function. A number of researchers have used complexity theory to show that feedforward networks with only two active layers (in their terminology, *three* layer networks) are universal approximators [Cybenko89], [Hornik89], [Hornik90], [White90]. Being a universal approximator means that by increasing the number of nodes in each layer, the networks can approximate any continuous function $\mathcal{F}$: $\mathbf{R}^M \rightarrow \mathbf{R}^R$. These results are proven for activation or "squashing" functions which are only constrained to be monotonic, increasing, continuous-valued but the rate of approximation is left as an open question.

Sontag has explored the capabilities of feedforward networks in terms of ability to classify and to interpolate by formal analysis of the ability of a network to form dichotomies of pattern space, and by measuring the Vapnik-Chervonenkis (VP) dimension of the architecture [Sontag90], [Sontag91]. The VP dimension is the cardinality of the largest set that can be arbitrarily partitioned using a particular network structure. Sontag's analysis compares relative abilities of one-hidden-layer networks with and without direct connections from input to output layer nodes and the abilities of networks using Heaviside vs sigmoid activation functions. The

[1] Heavisides do not have continuous first derivatives and are therefore unsuitable for use with back propagation.

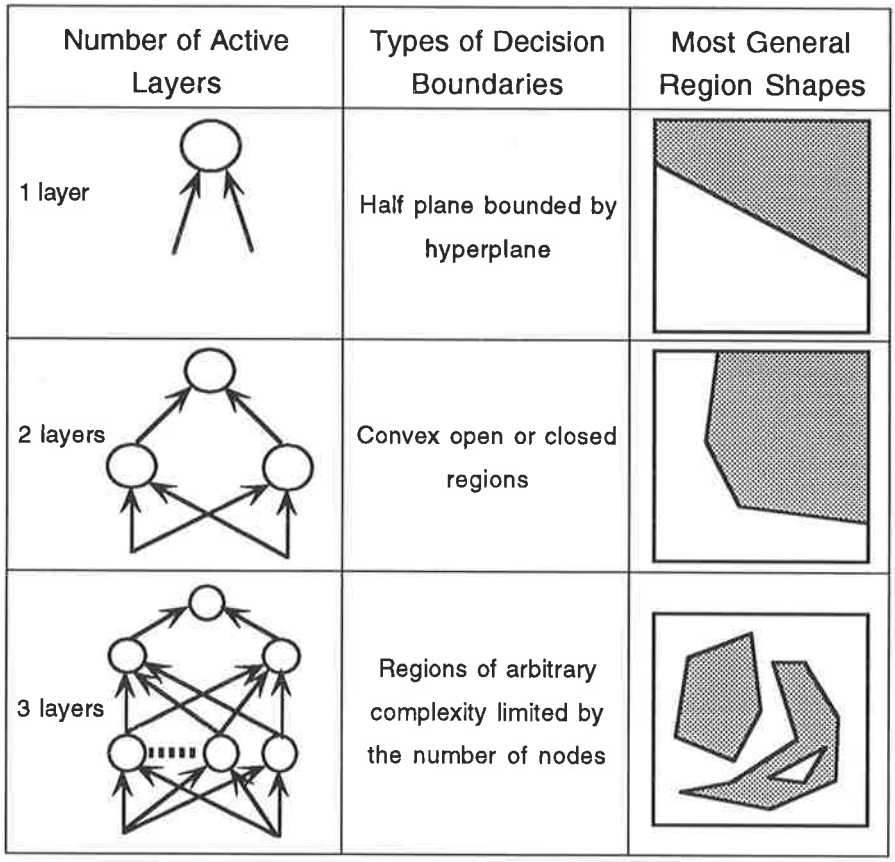


Figure 3.6: Input space partitioning by MLP networks with Heaviside activation functions (after [Lippmann87]).

separate comparisons show sigmoids and direct connections double the power of a particular architecture, but no results are given for combinations of conditions.

Levin *et al.* discuss a potentially useful result applying to training layered networks [Levin90]. They show by equivalencing minimum error and maximum likelihood training criteria, it is possible to derive, first a Gibbs distribution on the ensemble of networks with fixed architecture, and consequentially the probability of the correct prediction (classification) of a novel example. This probability is a formal measure of the network's generalization ability, but the application of this result is currently limited by the inability to evaluate certain required elements of the Gibbs function for practical cases.

It is fair to say that currently available results force an ANN designer to make many educated guesses about the overall MLP structure. Despite the naïvity of the original argument of Lippmann and the very thorough argument that one hidden layer is enough, there remains some distinct advantages in having more than the theoretical minimum number of layers. Sontag gives a general result showing that nonlinear control systems can be stabilized using two-hidden-layer networks, but not using one-hidden-layer networks [Sontag90a]. Lang and Witbrock [Lang88], Sietsma and Dow [Sietsma91] and others have observed that, in practice, an MLP with more than the minimum number of layers can be easier to train. The practical designer of an MLP should thus be able to take advantage of this reverse-of-the-curse-of-

dimensionality rule of thumb: add layers to decrease the number of required nodes (possibly exponentially). Disappointingly, Sietsma and Dow found that having fewer nodes arranged in more layers, and in particular having fewer nodes in the first layer, interfered with the ANNs ability to generalize.

3.2.5 Processing MSS data using MLPs

The procedure to classify a multi-spectral image can easily be cast as a neural net classification task. The training phase replaces the parameter estimation phase and there is no requirement for a (guessed) model of the classes' probability distribution functions.

Let the classes be represented by ω_r, $r = 1, \ldots R$. The MLP can be used as an N dimensional classifier by using N inputs and $\log_2(R)$ outputs. More commonly, R outputs are used, with the intention that when the input is classified as coming from class ω_r, the r th output is true (high, +1) and the other outputs are false (low, 0). In practice, where the output values of the function f are in some range, for example (0,1), the output having the largest value is used to determine the class of the input.

During training, for each class r, the training set for that class is used to teach the network to have the rth output = 1, and all other outputs = 0. The back propagation algorithm is used to adjust weights. During the classification phase, for each pixel, the elements of the pattern vector form the inputs and the pixel is classified according to the rule

$$\boldsymbol{x}_0 \in \omega_r \Leftrightarrow x_{L,r} > x_{L,s} \;\; \forall\, r,s = 1 \ldots R,\; r \neq s \tag{3.17}$$

(c.f. 1.10) where $\boldsymbol{x}_0$, in this context, is the vector of inputs regarded as the state vector for the zero'th layer.

This form of MLP is a special case of the theoretically analyzed MLP above requiring that the output be a (binary) vector. Given that the result on universal approximators, we view the network as defining R characteristic functions, one for each class

$$\Psi_r(\boldsymbol{x}) = \begin{cases} 1 \Leftrightarrow \boldsymbol{x} \in \omega_r \\ 0 \Leftrightarrow \boldsymbol{x} \notin \omega_r \end{cases} \tag{3.18}$$

and create R MLPs capable of approximating the characteristic function of each of R classes. These may be conceptually formed into one MLP in which the weights from nodes in adjacent sub-MLPs are all zero. In practice, there will be nodes that can serve more than one sub-MLP and integration of the sub-MLPs occurs without loss of function.

At this point it is useful to note that Minsky and Papert's prime interest in the perceptron was for its use in the recognition of structures in images [Minsky88]. At a micro level the structure of spatial primitives within a pixel's neighbourhood (of the kind described in Chapter 2) is texture. This approach is developed further in Chapter 4 by including this texture in the input layer.

3.3 ANN implementation

3.3.1 The problem domain

The aim of the experiments described here was to perform a supervised classification of four band multi-spectral image data into one of five classes. Five 20x20 pixel reference sets were abstracted from a 4-band Landsat MSS sub-image of the "Adelaide" scene taken March 1985. Figure 2.9 shows the Landsat sub-image used for most experiments and with the reference areas outlined. Each reference set was divided into a training set and a test set of 200 pixels. The training set was used to estimate parameters for ML classification, and also to train neural networks by example. The image data were quantized to 8 bits but were treated as continuous values which were scaled to lie in the range (0,1.0) for input to the neural networks.

Because of the way in which the training sets are selected, the probability that the elements of the training set come from the class they represent is less than (but hopefully near) one. This represents an unavoidable source of noise in the training data. It is not possible to establish criteria for the rejection of suspected noisy data points without making assumptions about the distributions of values. Indeed, Longstaff and Reid [Longstaff89] found that noisy training data was beneficial in avoiding local minima in weight space while training a network using gradient descent. A true gradient descent method cannot escape from local minima during optimization, but where weights are adjusted after each presentation of an input/output training pair, the direction of each adjustment in weight space will depend on the individual data pair. It was shown that in practice, when the adjustments based on different training data pairs all point in the same direction towards a local minimum adding noise can cure the problem. Where the path in weight space is determined by the ensemble of training data, this mechanism is not available since the characteristics of the ensemble do not alter from one weight adjusting pass to the next.

While Longstaff and Reid investigated a simple boolean logic problem, this thesis argues that the same reasoning can be applied to MLPs with continuous valued inputs and a collection of training data which is a representative sample from an unknown but smooth probability distribution. Experiments here showed that the inability to reach a desired error criterion could be cured in most cases by increasing the network size. With a trainable-sized MLP local minima were not observed despite a large number of trials. Conversely, in those cases where it was not possible to find a trainable-sized MLP, the large stopping errors were consistent for different weight initializations indicating that the network had found the, albeit unacceptably large, global minimum.

3.3.2 Implementation issues

The parameters to be chosen for these experiments were the number of nodes in each layer in the network, the value of learning rate η and momentum α (training parameters) and an appropriate stop-training criterion. The criteria were that the network had to converge in a reasonable time (preferably as fast as possible), and that the resulting accuracy of classification as measured by "number correctly classified" must be as large as possible. A major problem

in implementing the simulated ANN was the lack of clear implementation guidelines in the literature. It was particularly noted that most theoretical papers relied on small scale problems to show application of results. The XOR problem is seminal to the work of training MLPs using back-propagation and is an interesting and difficult problem for machines with simple nodes whose only function is thresholded sum-of-weighted-inputs. However, its results do not scale, and they are not necessarily applicable in systems where the inputs are continuous-valued instead of Boolean. At the time of beginning this work, published examples of systems with continuous valued inputs was limited to Gorman and Sejnowski [Gorman88], describing a limited domain problem. In the absence of descriptions of similar systems in the literature, parameters were determined pragmatically by experiment and by over-design.

3.3.2.1 Structure

A three layer MLP with Heaviside activation functions is demonstrably able to perform classification of multivariate data and allows interpretation of the weights using Lippmann's simple MLP explanation. An MLP with sigmoidal activation functions is seen to be a more general case of the MLP using the Heaviside activation function, and should be capable of the same function with fewer layers. An MLP with 3 (active) layers was chosen for the classification of MSS data primarily because it is certain to be trainable.[1]

The structure is described by the string N_0-N_1-N_2-N_3 where N_L the number of nodes in layer L. In this chapter, the MLPs have full interconnection between layers; i.e. a node in layer 2 will have inputs from all nodes in layer 1. Such full interconnectivity may not be required for this task but at this time theoretical results regarding the so-called fan-in of a node exist only for the case of boolean functions.(e.g. see [Redding91]). In subsequent chapters, full interconnection will not be assumed, and structure of interlayer interconnection will also need to be considered.

The values of N_0 and N_3 are determined by the input dimensionality and number of classes respectively, leaving N_1 and N_2 to be selected to give a network that was trainable, rather than optimal (figure 3.7). In the MLP the only information-carrying quantities are the weights. If we consider the weights as parameters which are to be determined by the data, it is possible to argue that training a ANN is an parameter estimation problem and it may be either *under-determined* or *over-determined.* This can be used as a guide to the MLP structure.

Following estimation theory, an MLP is considered to be fully determined if the number of weights in the network is equal to the number of *independent* elements in the (complete) training set, and under-determined if there are fewer weights. However, it is important to note that typical training sets contain much replication of data points. The replication conveys information about the frequency distribution of the data, and hence the importance to be given to each component of the data to be classified. Hence more weights may be justified. On the other hand, it is possible to model or "learn" P data points exactly, given P+1 degrees of

---

[1] The time taken for simulation was a peripheral but practically important issue. While not proven for scaled up networks, for "toy" problems a trainable 3 layer net will have fewer nodes than a 2 layer net, and hence need less time to simulate.

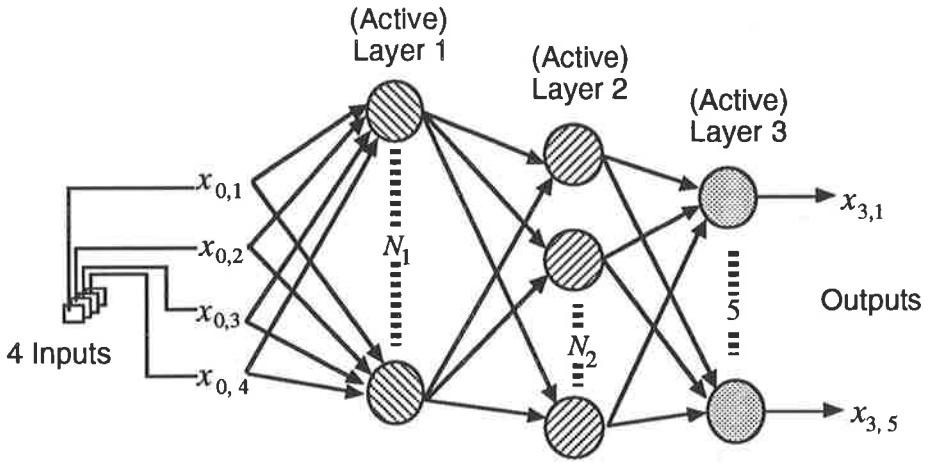


Figure 3.7: MLP structure.

freedom. Therefore an MLP is over-determined if it has more weights than there are training set elements and it is wise to set a (loose) upper bound on the number of weights to be the total number of training set elements.

There is a further expectation that a trained classifier should be able to generalize. In this problem domain, generalization is defined as the ability to classify unseen examples that differ only slightly from the elements of the original training set by distortion, noise or the natural variability of the ground reflectance. If there are enough free parameters to enable the neural network to learn every training data point exactly, then by analogy with polynomial curve fitting, there is no guaranteed predictable behaviour in any part of the input space not defined by training data. Hence a criterion for the selection of ANN structure is that it should *not* be capable of memorising every point in the training set. An ANN approach to a problem is valuable if the use of nodes with non-linearities makes it possible to capture the nature of the training data, to the same degree of accuracy, with fewer parameters that a classical system.[1]

In order to obtain an empirical guide to the influence of structure on the classification result, 175 different MLP structures of type 4-N_1-N_2-5 were trained over 50,000 iterations to recognize the 5 classes of training data. There were 200 examples of each class in the training set making a total of 1000 examples. The classification accuracy of each trained structure was evaluated by classifying a test set of data which was not the same as the training set. To overcome any dependence on the random initialization of the weights, 10 trials for each structure were carried out with different initializations. Weights were initialized with values from a uniform distribution with the range (-1.0, +1.0).

There are a variety of ways of presenting the results and hence influencing the interpretation. For an MLP with insufficient nodes, common mode of failure is for it to ignore the existence of

[1] In fact, there can be no reason to use ANNs if there is not a possible more efficient recoding of the information via the non-linearities, which here serve as basis functions. A good reason for relying on the sigmoid non-linearity to provide the generalization capacity is that it is biologically inspired. (Currently the human brain is considered to be the best available generalization system.)

one class and learn to be as good as possible on the other classes[1]. This behaviour produces a worthless neural network, but one that may have an apparently acceptable *mse* calculated using (3.4) or its variant used in the next section. For these tests of structure, the error analysed was the largest error made in classifying any of the 5 classes; i.e. the "maximum class error" (MCE). This number can range from zero for all test data correctly classified to 100% indicating one or more classes completely misclassified.

It is also not straightforward to measure useful statistics that summarise the results from 10 trials. If classifications will based on a single trainable MLP rather than the most trainable of a selection of MLPs, the expected value of the error over the 10 trials is the most useful value. If an MLP is selected on the basis of the best of 10 trials (which must be assessed using an auxiliary test set), then it is valid to look at the minimum value over all trials. In this section, the expectation is examined, but elsewhere it will be relevant to adopt the second approach.

To calculate the expected value of MCE an estimate of the probability distribution function is required. In the absence of extensive trials for each structure, the best estimate is of Gaussian error, so we examine average value and the standard deviation. Figure 3.8a represents the MCE for each structure averaged over 10 trials. Figure 3.8b shows the standard deviation calculated over the 10 trials for each structure to provide some information on the variability of the result. Student's *t*-distribution for a small sample of 10 trials gives a 99% confidence interval of approximately 1.25 times the standard deviation [Freund71]. Here, for the non-100% error cases a typical value of standard deviation is 1.5%, which gives a 99% confidence interval of ±1.9%

An approach to training a network discussed in section 3.2.5 is to train a number of ANNs to learn a sub-goal and then place the ANNs along side each other to effectively become a multi-class classifying ANN. There is an inherent belief that some of the nodes will be able to be shared hence forming an integrated network. The size of the MLPs found useful for 5 class classification of MSS data was relatively small. Since there is no guide to which nodes can be shared before training, it would be necessary to train sub-nets with the same number of nodes as the complete network in all but the last layer, and then prune the network by eliminating and sharing nodes. For this particular classification task there is no obvious benefit accruing from the sub-goal approach.

[1] This is rather like a student electing to only learn certain parts of a subject in the hope that only those sections will be examined

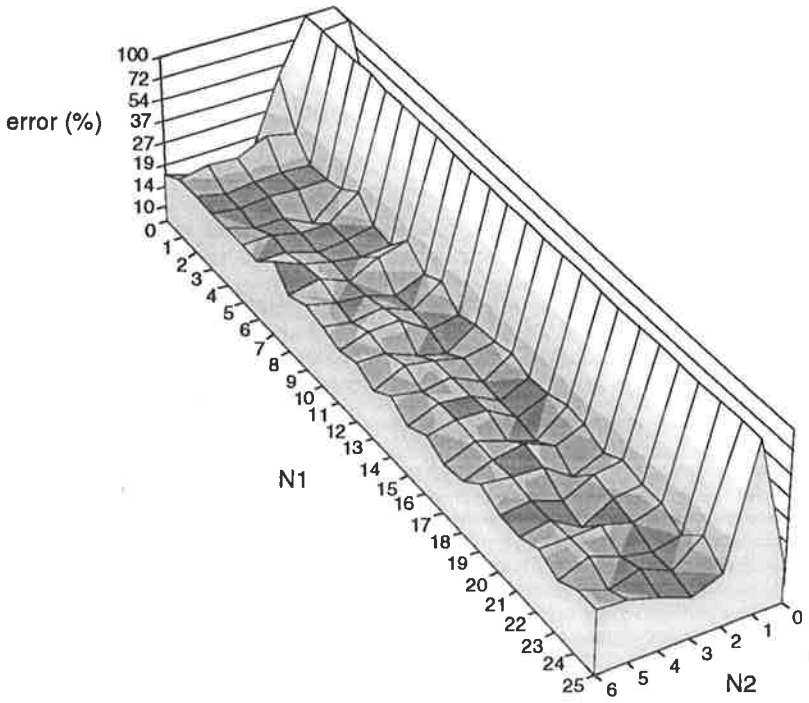


Figure 3.8a: MCE averaged over 10 trials for MLPs with structures 4-N1-N2-5.

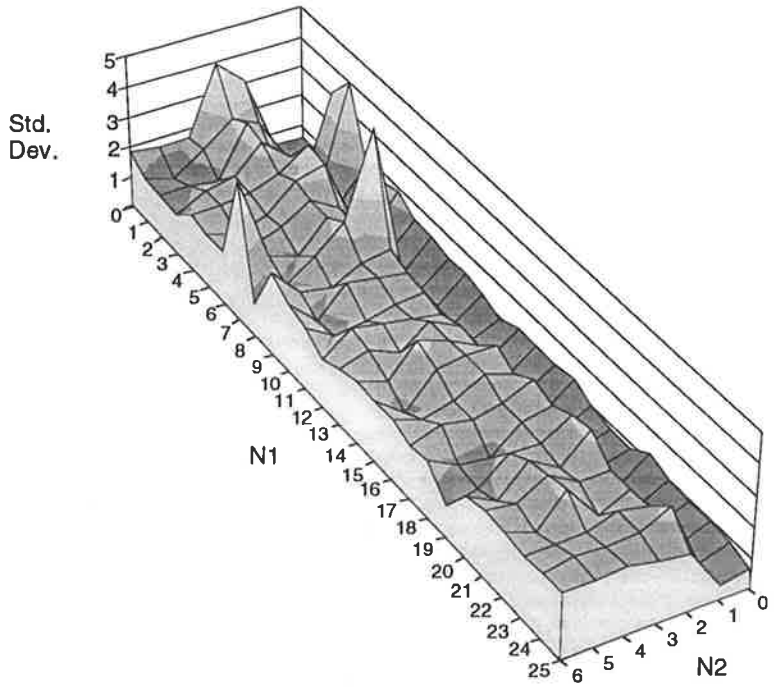


Figure 3.8b: Standard deviation over 10 trials of MCE for MLPs with structures 4-N1-N2-5.

Figure 3.8 suggests that a 4-0-6-5 MLP with a single hidden layer can *on average* produce a result at least as good as a two-hidden-layer network. While this is shown more clearly in figure 3.9a, figure 3.9b shows that the best result achieved over 10 trials was always obtained by an MLP with two hidden layers.

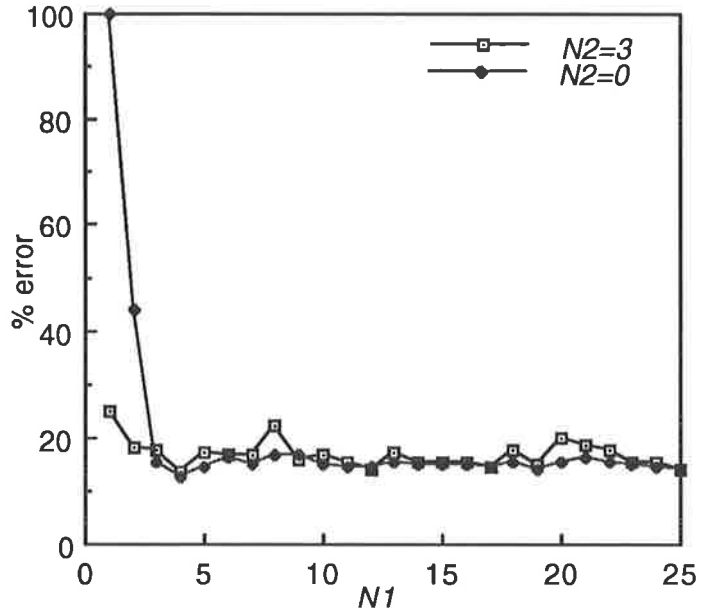


Figure 3.9a: Average MCE for MLPs with structures 4-*N*1-3-5 and 4-*N*1-5.

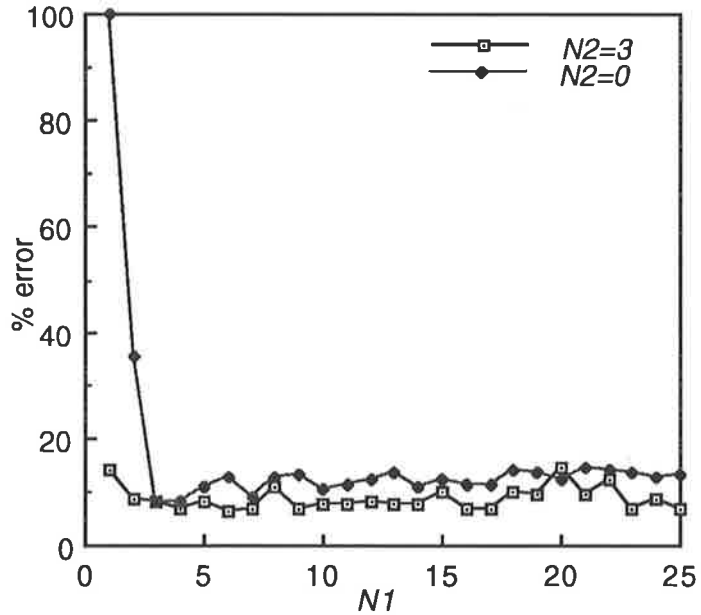


Figure 3.9b: Minimum over 10 trials of MCE for MLPs with structures 4-*N*1-3-5 and 4-*N*1-5.

3.3.2.2 Learning rate (gain) and momentum

The learning rate η and momentum α were selected by experiment to be 0.3 and 0.0 respectively. Some authors(e.g. see [Tollenaere90]) have suggested that it can be beneficial, even though computationally expensive, to choose the parameters to be specific for each node. In practice, such a scheme is difficult to manage and in the experiments described here η and

were chosen to be common to all nodes. Figure 3.10 is a plot of gain vs mean square learning error (*mse*), as defined by (3.4).

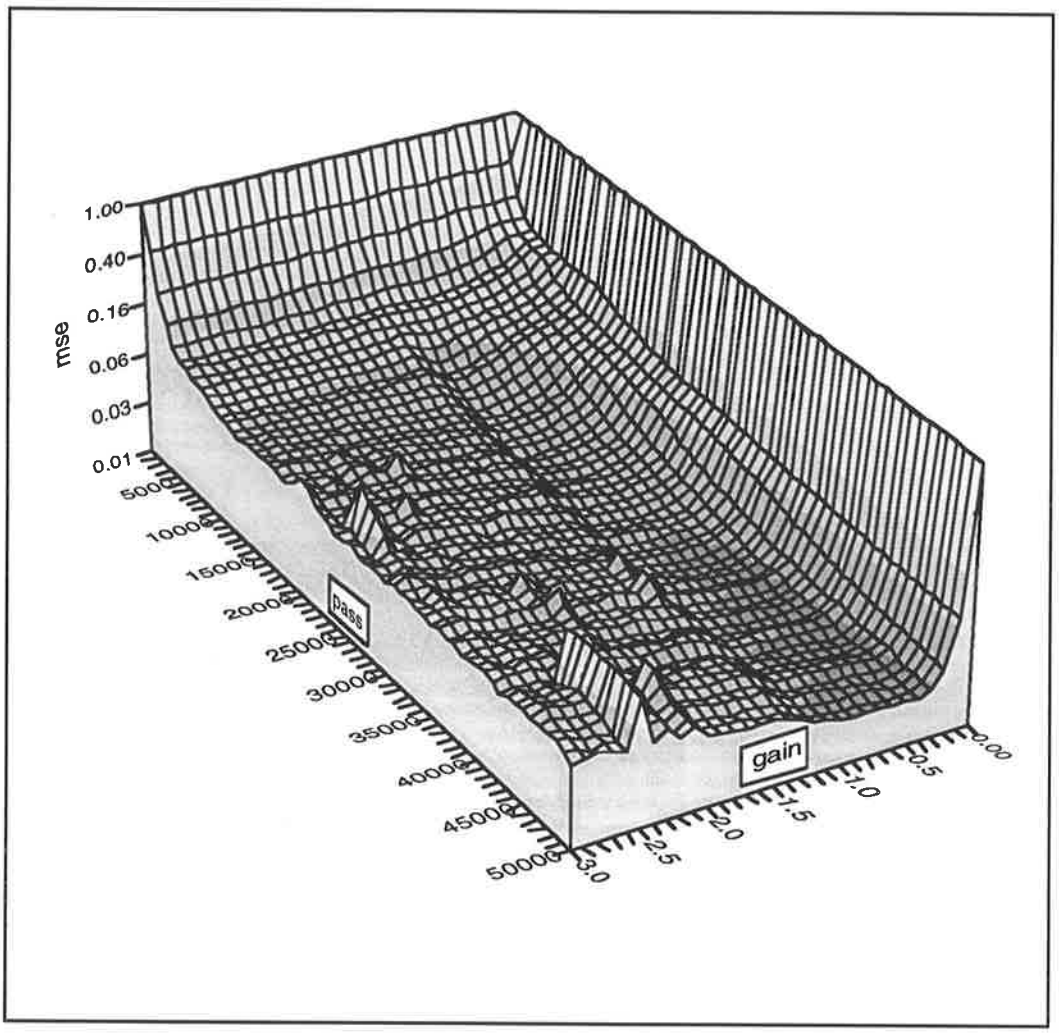


Figure 3.10: *mse* vs iteration with different gains for a 4-5-6-5 MLP.

Figure 3.11 is a section across the end of plots of the kind shown in figure 3.10 for 3 different structures of MLP. Consistent with the theory, larger gains are more prone to error due to taking over-large steps in weight space during gradient descent. Smaller gains are favoured but choosing a gain too small causes no learning to take place.

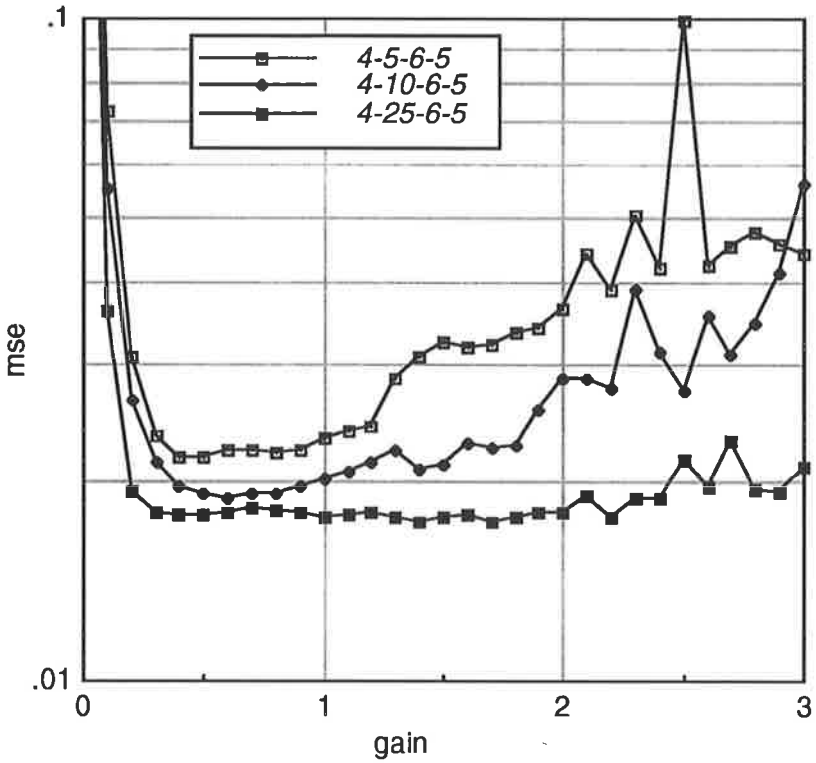


Figure 3.11: *mse* vs gain after 50,000 iterations for MLPs of structure 4-N1-N2-5.

3.3.2.3 Stop-Training criterion

The stop-training criterion is pivotal to the optimal training of the MLP. The most obvious criterion is that for some $\varepsilon \approx 0.1$ (10% of max value)

$$s_{\boldsymbol{x}} < \varepsilon \quad \forall\ \boldsymbol{x} \in \mathbf{P} \tag{3.19}$$

where $s_{\boldsymbol{x}}$ is the training error for a particular training input $\boldsymbol{x}$, and $\mathbf{P}$ is the set of all training patterns. Evaluation of this criterion can only take place at the end of each epoch, which is not satisfactory when adjusting the weights after each pattern. (Here an epoch is a complete pass through the training set). An error criterion which may be evaluated with minor overhead after each weight adjustment is defined by

$$s_x^k < \varepsilon \tag{3.20}$$

where s_x^k is the smoothed mean square training error, defined as

$$s_x^k = \frac{1}{P}\{(P-1)\, s_x^{k-1} + \sum_{i=1}^{M} (x_{Li}^{\mathbf{P}} - d_i^{\mathbf{P}})^2\} \tag{3.21}$$

and s_x^{k-1} is the previous value of s_x^k and P is the total number of patterns in the training set. s_x^k is initialized with $s_x^0 = 1.0$. This pessimistic but useful definition is required because of the "noisy" nature of the training sets.

An alternative attractive stop-training criterion is to stop when the benefit gained per weight adjustment (also called benefit per iteration) falls below a given threshold. Figure 3.12 shows the training histories for three experiments up to 50,000 weight adjustments. The near-straight

line plotted on a log-log axis suggests the *mse* is a function of the iteration, k (and hence of the epoch) of the form

$$s_x^k = c\, k^a \tag{3.22}$$

where c and a are constants. As the weights are refined to allow the ANN to approximate the required learning rule more closely, gains in *mse* are masked by individual classification errors especially from outliers in the training data, giving a "noisy" curve, and making it hard to estimate the benefit per iteration. By fitting a curve of the form of (3.22) to *mse* vs iteration data it is possible to predict when the benefit will fall below the threshold. A similar metric used in optimization is the relative error stopping criterion which has the form

$$\frac{\left|s_x^k - s_x^{k-1}\right|}{s_x^k} < \varepsilon \tag{3.22}$$

Clearly this involves storing a complete set of s_x^k, one after each presentation of a pattern. A further related stopping criterion would require keeping a running average of the size of Δw and stopping when $\frac{|\Delta w|}{|w|}$ was sufficiently small.

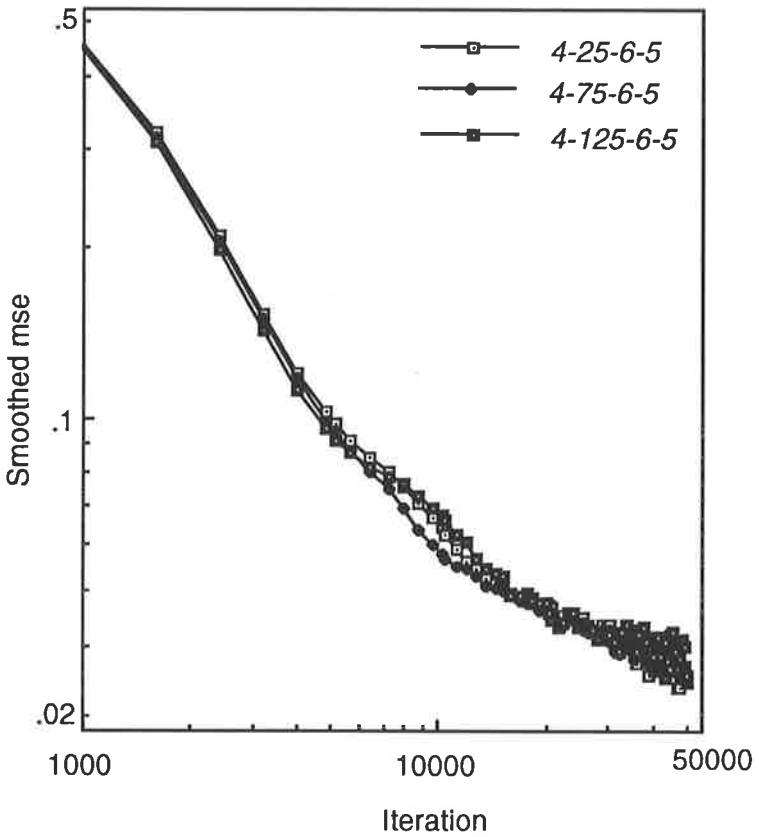


Figure 3.12: *mse* vs iteration for different MLP structures.

Another possible stopping criterion is *cease training just before "over-training" occurs.* "Over-training" is defined to begin when further adjustments of the weights to minimize *mse* cause an increased error in classification of an independent training/test set [Chui91], [Marshall91] . By analogy with polynomial curve fitting, if there are more degrees of freedom than required (here, more weights than required) the network can learn the test data exactly.

However, because of the extra degrees of freedom, there are no constraints on the behaviour of the function for data near the fitted points and hence no guaranteed powers of interpolation or generalization .

While there can be no argument with the concept, the practical application of this criterion is flawed. It is necessary to have an independent auxiliary training/test set to assess whether the net is trained. The independent data in the auxiliary set could have been used to give a better estimate of the weights and hence avoid unpredictable behaviour for data from that particular auxiliary set. Testing separately against the auxiliary set can only guarantee that the network will behave predictably for a particular combination of test and auxiliary set. A positive aspect of the auxiliary dataset approach is that training is faster. However, the accuracy achievable is bounded above by the accuracy of the combined-dataset case and may be significantly less.

In practice, the only usable stopping-training criteria was found to be smoothed mse modified to include a maximum number of weight adjustments, since in most cases, the maximum training error of 0.1 could not be reached. Kolen and Goel [Kolen91] call this t-convergence, where t is the limit on the number of epochs. Most results in this thesis indicate accuracies after 50,000 weight adjustments, which appeared to be a good compromise stopping time. Other trials carried out with the same data verified that there was no benefit in increasing this number to 100,000.

3.3.3 Analysis of best classification accuracy.

In this section we investigate the results of accepting the network structure and set of weights that produces the best classification accuracy (least error) over 10 training sessions. This leads to a different view of the results of the trials in which 175 different MLP structures were tested. The graph of minimum MCE over 10 trials (figure 3.9b) shows that while a one-hidden-layer network may be appropriate for a situation in which only one try at learning is allowed, where a number of trials are possible the two-hidden-layer network is capable of a more accurate result.

Considering only two-hidden-layer networks, figure 3.13 shows a surface plot of minimum MCE over 10 trials for hidden layers up to size 25-6. The results for 1 node in the penultimate layer are not shown since such nets all scored 100% error. A simple conclusion from this diagram is that between 3 and 6 nodes in each of the two hidden layers will give a good result.

Figure 3.14 shows summary statistics and a percentile plot of the MCE values encountered over the 1750 trials. The network with structure 4-6-3-5, gave the minimum value of 6.5% MCE. Such a network has 67 adjustable parameters (weights) There were a further 11 instances giving error of 7%.

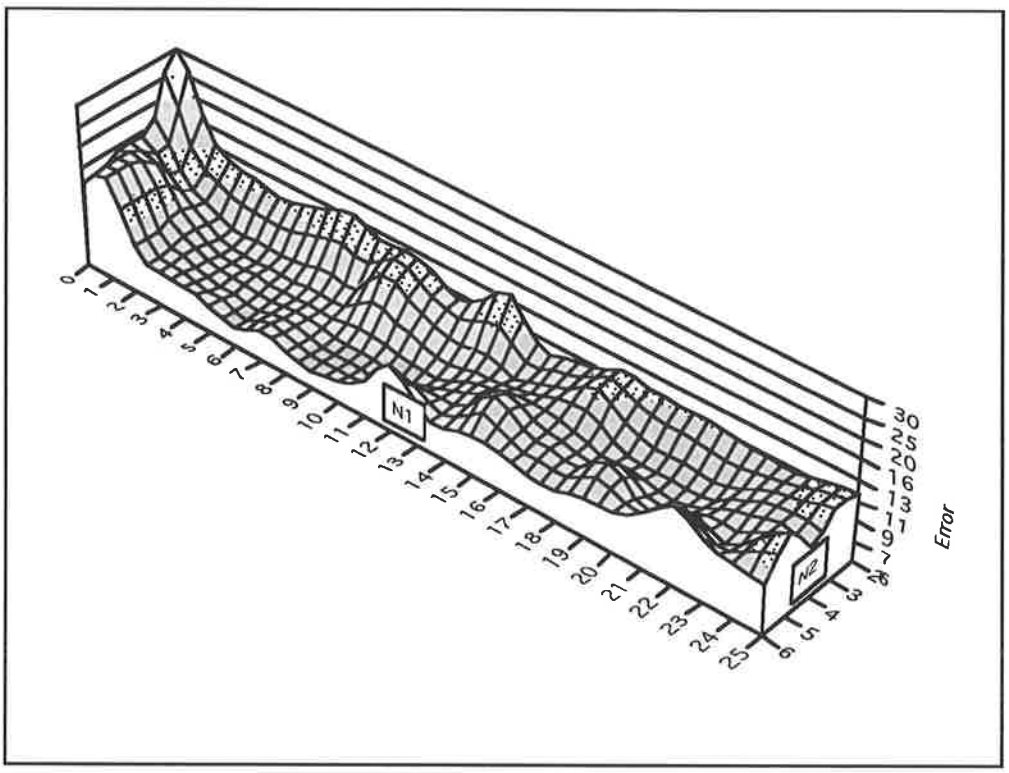


Figure 3.13: Best over ten trials of max class error after 50000 iterations using different network structures 4-*N*1-*N*2-5.

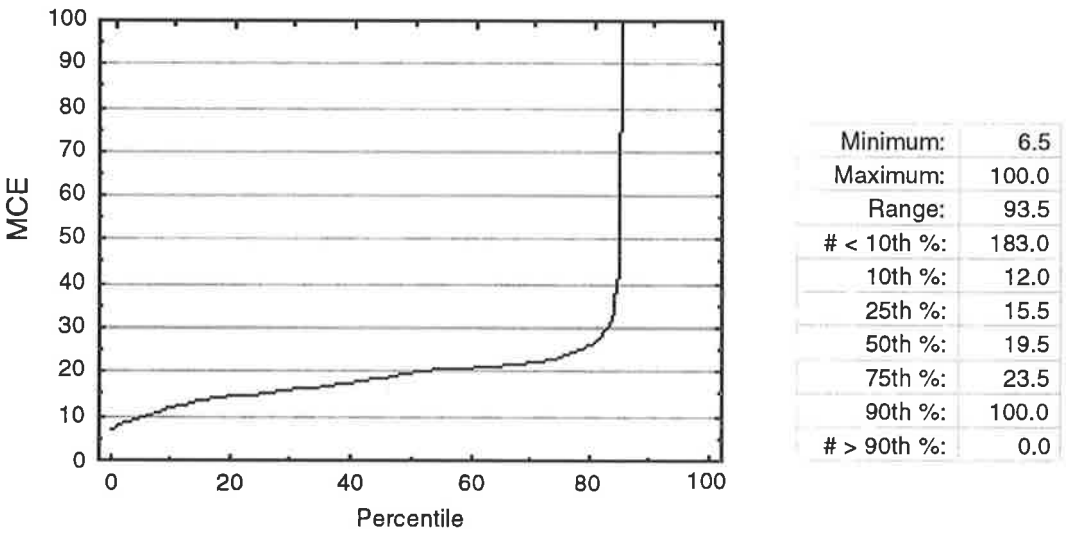


Figure 3.14: MCE over 10 trials of each of 175 MLP structures- some structures with 100% error or with no nodes in one hidden layer omitted for clarity.

The MLP classification examples with the minimum MCE are presented in table 3.1. For the purposes of comparison, classification accuracy from standard ML classification using the same training and test sets is included as a benchmark. In each of these selected best-cases, the MLP is able to achieve much better accuracy than the ML benchmark.

Table 3.1: Comparison of classification accuracy

Classifier	Class Errors (CE) (%)					MCE (%)	ACE (%)
	1	2	3	4	5		
ML	0.0	0.0	16.0	6.9	17.4	17.4	8.1
MLP: 4-6-3-5	0.0	1.5	6.5	6.0	6.5	6.5	4.1
MLP: 4-3-5-5	0.0	1.5	7.0	7.0	7.0	7.0	4.5
MLP: 4-4-3-5	0.0	1.5	7.0	7.0	6.5	7.0	4.4
MLP: 4-7-3-5	0.0	1.5	7.0	6.5	6.5	7.0	4.3
MLP: 4-25-3-5	0.0	1.5	6.5	6.0	7.0	7.0	4.2
MLP: 4-3-4-5	0.0	1.5	7.0	7.0	4.0	7.0	3.9
MLP: 4-17-3-5	0.0	1.5	7.0	6.0	6.5	7.0	4.2
MLP: 4-4-5-5	0.0	1.0	4.5	7.0	6.0	7.0	3.7
MLP: 4-16-3-5	0.0	1.5	6.0	7.0	7.0	7.0	4.3
MLP: 4-9-3-5	0.0	0.0	6.5	7.0	5.0	7.0	3.7
MLP: 4-21-4-5	0.0	1.0	6.5	7.0	7.0	7.0	4.3
MLP: 4-23-3-5	0.0	1.5	6.0	5.5	7.0	7.0	4.0
MLP: 4-25-6-5	0.0	1.5	9.5	4.5	10.0	10.0	5.1

As intimated above, while for a particular classification task the average class error (ACE) is of some interest, the maximum error for a class (MCE) is of considerable practical importance. Further, because of the non-linear nature of the ANN classifier the two quantities are not linked by a simple functional relationship.

3.3.4 An example of real image classification

Figures 3.15 and 3.16 show the classification of a 512 line by 640 column subset of the "Adelaide" scene used for the generation of the reference sets used above. The five 20x20 pixel reference sets used in the previous section were used to estimate parameters for ML classification, and also to train the neural network.

The accuracy of classifying real images is difficult to measure without 100% ground-truth data. However, we observe that pictures are similar but that they contain a number of errors which depend on the nature of the classifier. The ML technique makes errors where classes have texture. ML also fails where the training set was not diverse enough to describe all possible members of that class; i.e. it is not able to generalize. The MLP fails for the same reason but on different image features, in particular, the area of cloud shadow. Neither the ML nor MLP classifiers were presented with any data from the area of cloud shadow during training and the substantially correctly classification of cloud shadow by ML emphasises the robustness of the conventional technique.

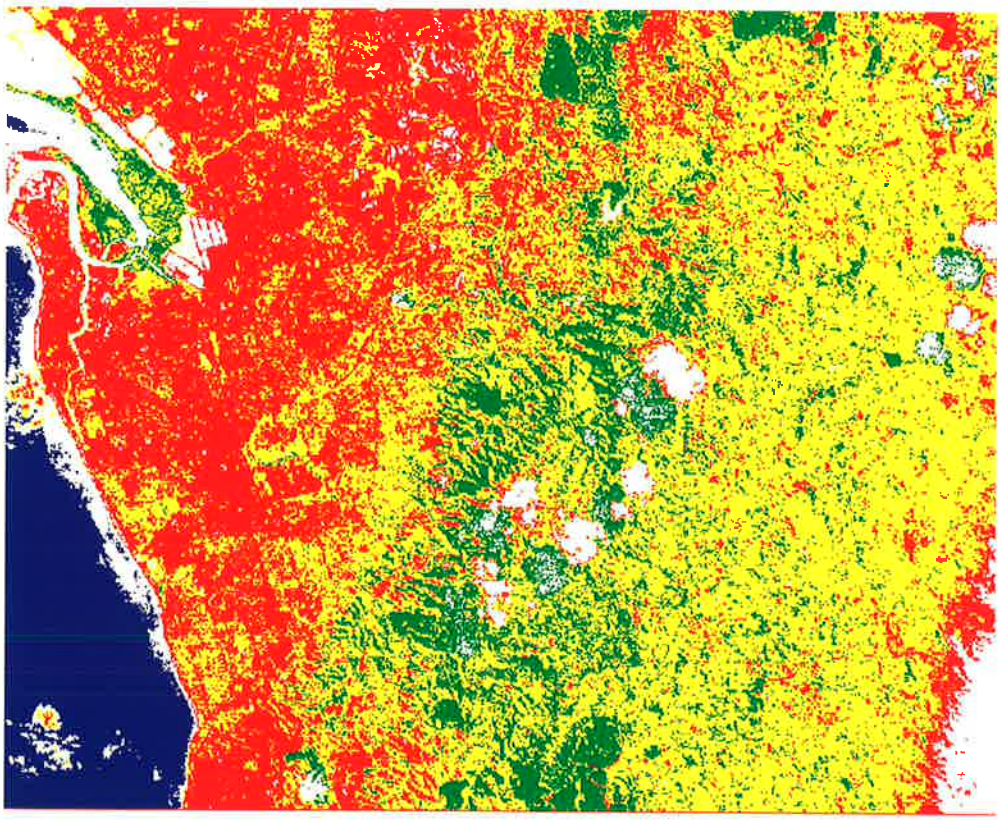


Figure 3.15: Classification of an MSS image using ML

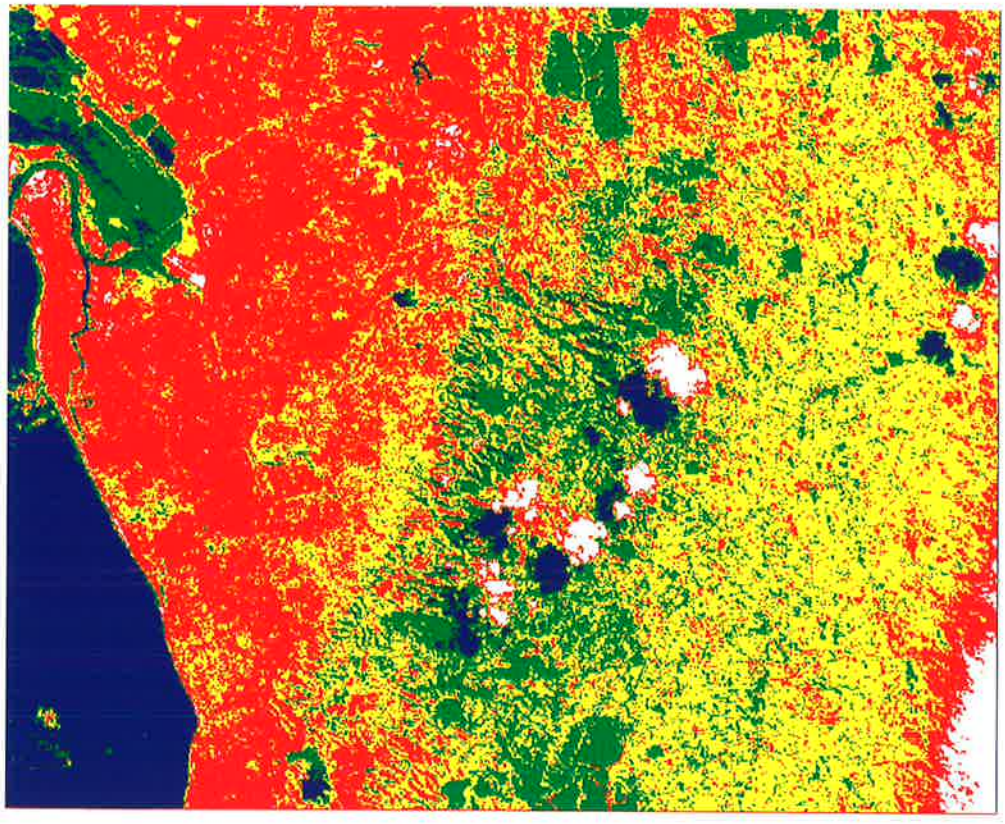


Figure 3.16: Classification of an MSS image using MLP
Blue=water, white=cloud, red=urban, green= trees, and yellow=farm

3.4 Analysis

3.4.1 Overview

Given that it is possible to train an MLP to distinguish multi-spectral classes with an accuracy approximating that of ML, an interesting question is "can we look inside the ANN to see what it learnt". One approach has been to create connectionist expert systems in which the internal representation is imposed on the neural network as part of the training. This approach via synthesis is not easily inverted for analysis. Rules explaining the behaviour of nodes are accessible only in the simplest of networks [Gallant88], [Sun91].

Since the weights in the network are the only information storing mechanisms it seems obvious to see if they contain a recognisable form or the support for a recognisable model. Section 3.4.2 presents a brief examination of the trained weight values from a trained MLP and makes some observations about structure. Section 3.4.3 presents a method for mapping the partitions of the input space formed by the weights by transforming the input space and then taking a two dimensional projection.

In 3.4.4 an attempt is made to examine the "energy" surface formed by each output node plotted against the input variables. This is informative for 2-dimensional inputs, but becomes difficult in higher dimension input space. A novel approach to mapping the nodal responses to 4 dimensional inputs is given. This approach has not been used before in this context.

Having explored the information content of the network , section 3.4.5 concludes with an attempt to reconcile the developed internal representation with the model of the MLP as a non-linear filter. There is also a postscript on the pruning of networks.

For the purposes of analysis, three trained MLPs were selected from those documented in table 3.1. These trained networks comprised the weight values for which best results were obtained for networks of structure 4-6-3-5, 4-3-5-5 and 4-25-6-5 . The actual weights are given in table 3.3 at the end of this chapter.

3.4.2 Interpreting raw weights

A number of researchers have attempted to assign importance to particular weights in a trained ANN. For example, in a number of publications Hinton has used boxes of shapes proportional to the size of each weight to produce a map of weights for interpretive purposes (e.g. see [Hinton87]). For an MLP, such as those used previously in this chapter, of any but trivial size, trained by a complex training set, it needs courage to single out any weight as being responsible for any particular behaviour. The final weights for one of the 4-25-6-5 MLPs trained in section 3.3.2 is represented as a colour coded image in figure 3.17. Observing this image, it is clear that certain nodes are not contributing much to the result. For example node 2 in layer 2 makes little difference to the output of the layer. (A weight near zero is represented by a colour close to white.) Stronger colours in the representation of layer 3 weights indicate that output nodes learn to make more defined decisions than the input nodes.

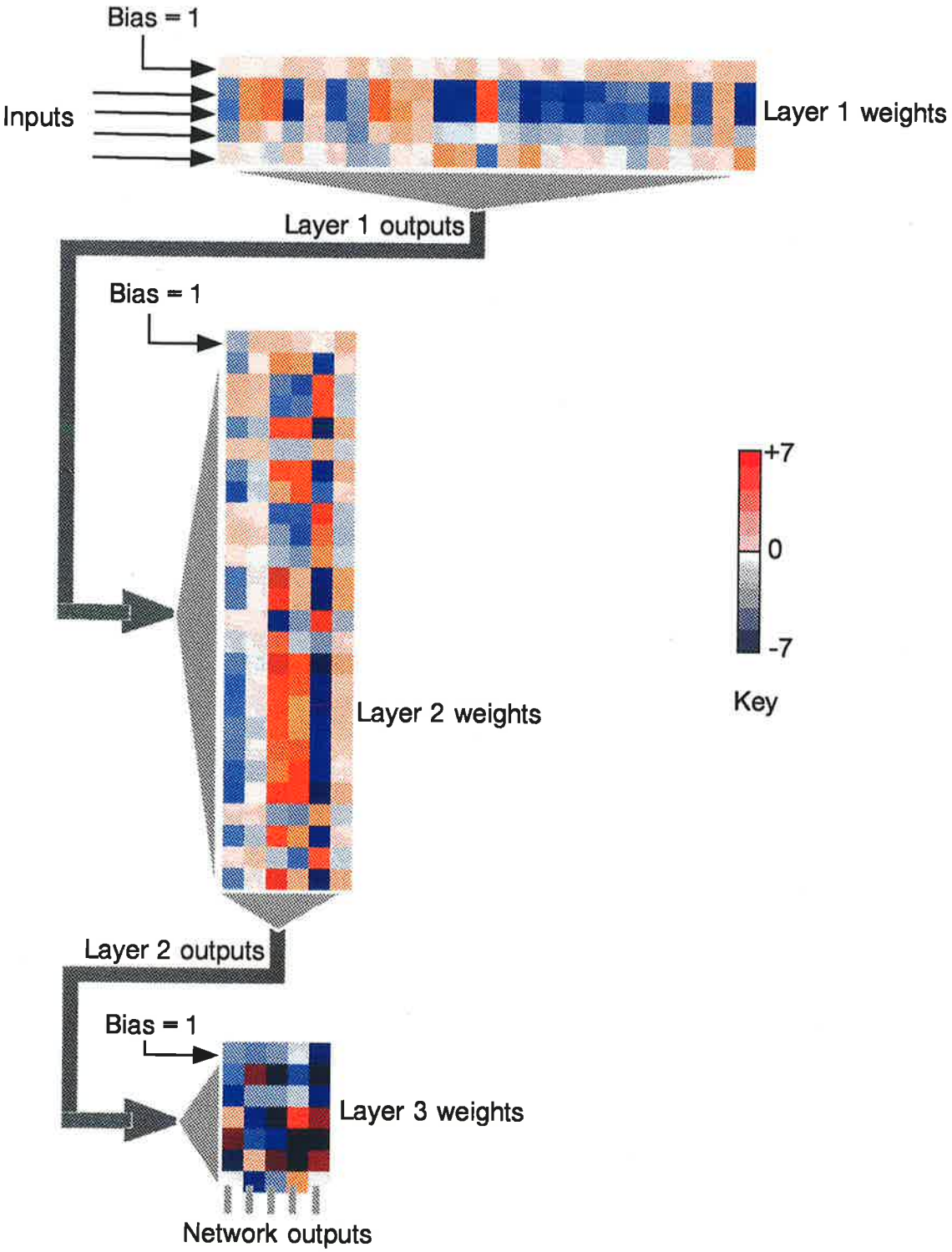


Figure 3.17: Final weights for an MLP trained to classify spectral data. Weight values are colour coded as indicated by the key. Weight diagrams are arranged to indicate flow of data through the MLP

While a static display of weights is difficult to interpret, observation of the development of the weights can lead to some fruitful conclusions. A dynamic display of the development of these weights is provided in attachment A. The dynamic display shows 68 frames representing the weights in a network being trained, sampled every 800 weight updates. Interesting features of the development of the weights are the obvious early development of the weights in layers nearest the output layer. Another remarkable feature is that none of the weights depicted changes sign during training. This suggests that the random initialization of the network weights is used as a basis for development, with correct weights being reinforced and incorrect weights simply being reduced to an insignificant value. This process would seem to have a human-like interpretation in that the ANN begins with prejudices, none of which are lost, but some of which are ignored while others are favoured. The fact that some connections become irrelevant after training has been recognized by Sietsma and Dow [Sietsma88] and has been used as one of the criteria for network pruning. A related algorithm for pruning in the light of activation maps is discussed below.

3.4.3 The partition of pattern hyperspace

In the explanation of the operation of an MLP by Lippmann [Lippmann87] the first active layer provides hyperplanes that partitions input space. When the activation functions are sigmoids instead of Heavisides this is a loose approximation. However, this explanation is intuitively appealing and is easily visualized when the input space is two dimensional. Since data here is four or more dimensional, a dimension reducing transformation is sought.[1]

We seek to visualize hyperplanes of the form of (3.1) which we rewrite here as:

$$\{x:\ w^{\mathrm{T}} x = 0\} \tag{3.23}$$

which, expanded for a 4-D input space becomes

$$w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 = 0$$

Data collected by multi-spectral sensors has large between band correlation, making eigen-vector analysis suitable. Based on the covariance of the input sample data, this form of feature reduction comprises a transformation of the data into a space in which most of the information is contained in a smaller number of features. This is variously known as principal component analysis, KL transform and singular value decomposition. Principal components analysis has been widely used for the analysis of MSS data [Jensen79], [Richards86], [Li89]. This transform is a rotation in input space of the normalized data, based on the eigen-vectors and eigen-values of the sample input data covariance. In matrix form the transform is written:-

$$Y = \Lambda^{-1/2} E^{\mathrm{T}} (X - \overline{X}) \tag{3.24}$$

where $Y, X \in \mathbf{R}^{4\mathbf{P}}$, $X = [x^1, \ldots x^{\mathrm{P}}]$ is the matrix of training data vectors, $\overline{X} = \overline{x}\,1_{\mathrm{P}}^{\mathrm{T}}$, where $1_{\mathrm{P}}^{\mathrm{T}} = [1, \ldots 1]$ is a vector of p ones, $\Lambda^{-1/2}$ is $\mathrm{diag}(\lambda_1^{-1/2}, \ldots \lambda_N^{-1/2})$, where λ_i are the eigen-values arranged in descending order and, E contains the eigen-vectors corresponding to the λ_i arranged in columns.

---

[1] It is possible to produce a representation in 3D, but the multiplicity of planes yields confusion.

For any particular vector, this gives

$$y = \Lambda^{-1/2} E^T (x - \bar{x}) \tag{3.25}$$

For convenience, rewrite the equation for the hyperplane given by (3.23) using un-augmented vectors

$$\{x: w^T x = -w_0\} \tag{3.26}$$

In the transformed input space, the hyperplane can be written

$$\{y: w^T (E \Lambda^{1/2} y + \bar{x}) = -w_0\} \tag{3.27}$$

$$\{y: w^T E \Lambda^{1/2} y = -w_0 - w^T \bar{x}\} \tag{3.28}$$

Writing the hyperplane in the new coordinate system as

$$\{y: v^T y = -v_0\}$$

it is possible to identify

$$v_0 = w_0 + w^T \bar{x} \tag{3.29}$$

$$v_i = \left[w^T E \Lambda^{1/2}\right]_i \tag{3.30}$$

$$v_i = \lambda_i^{1/2} E_i^T w \tag{3.31}$$

If only the two most significant axes are considered, then depending on the input data, often with only a minor loss of information, it is possible to display the data in 2D. The partitions of input space formed by nodes in the first active layer of an MLP can be shown in the new coordinate system by a line of the form

$$y_2 = a y_1 + b \tag{3.32}$$

with

$$a = -\frac{v_1}{v_2} \tag{3.33}$$

$$b = -\frac{v_0}{v_2} \tag{3.34}$$

Table 3.2 gives the numeric values calculated using the covariance matrix of the combined training sets (i.e. training data representing all classes). The large value for "variance proportion" indicates that for the combined training set it is possible to concentrate most of the information into the first principal component. However, this does not carry any information about the usefulness of the first principal component in distinguishing between classes. In this case, the transform is being used as a visualization tool.

Table 3.2a: Eigen-values and vectors of reference data

Eigen-values		Variance Proportion
λ_1	19835.58	0.984
λ_2	282.95	0.014
λ_3	30.83	0.002
λ_4	7.75	0.000

Eigen-vectors				
	$(*x_1)$	$(*x_2)$	$(*x_3)$	$(*x_4)$
E_1	-0.5	-0.532	-0.502	-0.464
E_2	0.526	0.412	-0.357	-0.653
E_3	-0.533	0.395	0.566	-0.49
E_4	-0.435	0.626	-0.548	0.344

Table 3.2b: Principal components transform based on covariance of reference data

$y_1 = -0.003551\ x_1 - 0.003778\ x_2 - 0.003561\ x_3 - 0.003294\ x_4 + 1.101749$
$y_2 = 0.031270\ x_1 + 0.024476\ x_2 - 0.021249\ x_3 - 0.038804\ x_4 + 0.674074$
$y_3 = -0.095937\ x_1 + 0.071091\ x_2 + 0.101914\ x_3 - 0.088277\ x_4 + 0.463162$
$y_4 = -0.156320\ x_1 + 0.224720\ x_2 - 0.196865\ x_3 + 0.123578\ x_4 + 1.720912$

Figure 3.18 shows the partitioning of the transformed input space by nodes in the first layer of each of the 3 pre-trained MLPs selected for close analysis. The lines represent the intersection of the transformed decision hyperplanes and the hyperplane containing the first and second principal components axes, y_1 and y_2 (with. y_3=0, y_4=0). In each case it is clear that the decision hyperplanes are oriented nearly at right-angles to the direction of the first principal component. Recall that the direction of the transformed axes is determined by the complete training data and not necessarily the best selection for separating a particular pair of classes. Therefore is possible and likely that some of the discriminatory power of the network is present in other principal components, explaining the fact that the decision planes' intersections with the y_1 axis are not exactly at right-angles. Maps of other y_i combinations did not give clear separation of input data into groups, with lines far from any data, probably because of a limitation of the representation. Even in figure 3.18 some decision hyperplanes appear to be in positions that will not contribute to a classification decision. This interpretation is a consequence of the limitation on these diagrams to depict *only* the intersection of a hyperplane with the axis. The output from a node is a continuous valued function due to the continuous valued activation function (the sigmoid) and hence contributes to the decision even when the plotted line is distant from the data point being classified.

Of most interest is the depiction of the 4-3-5-5 network, where it is necessary to be as efficient as possible in the first hidden layer in order to code the information for classification in just the 3 states which are passed to the next layer. Since there are only 3 nodes in the first active layer, it is possible to show the recoded data as a 3D scatter plot (figure 3.19). It can be seen from the plot that the classes have become separable, spread out over the possible locus of values allowed by the activation functions.

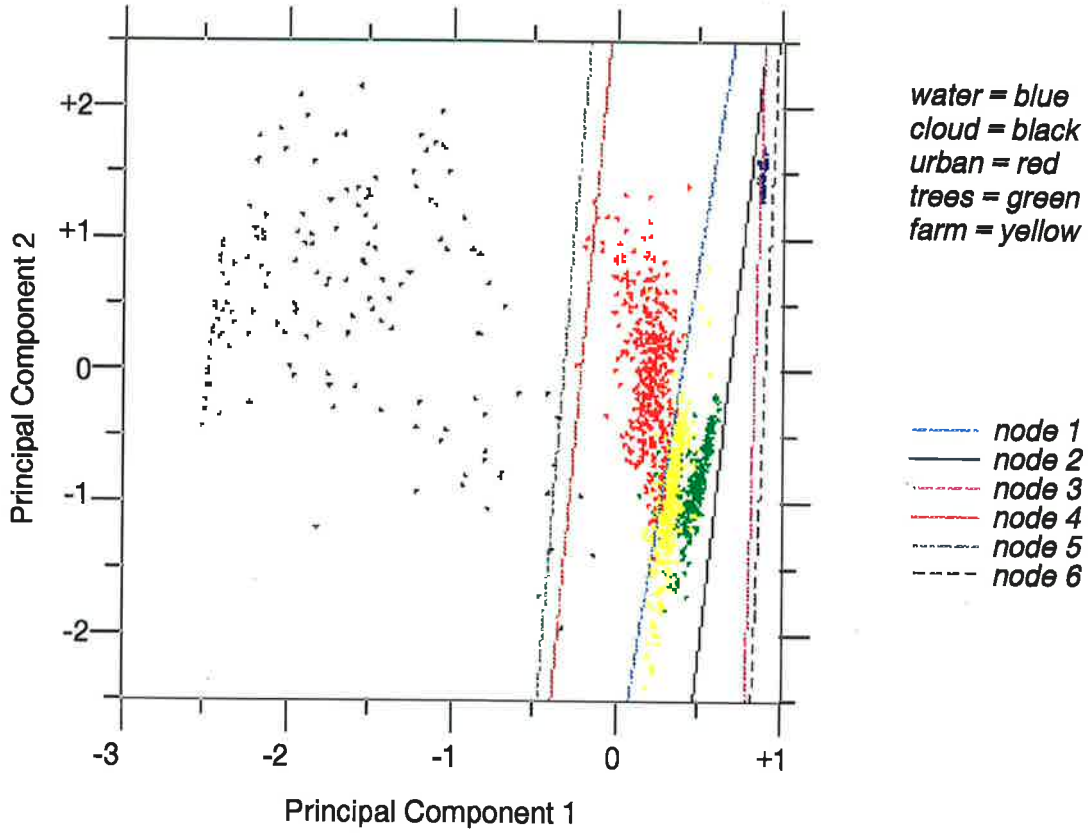


Figure 3.18a: Partitions of MSS data in PC space by MLP 4-6-3-5

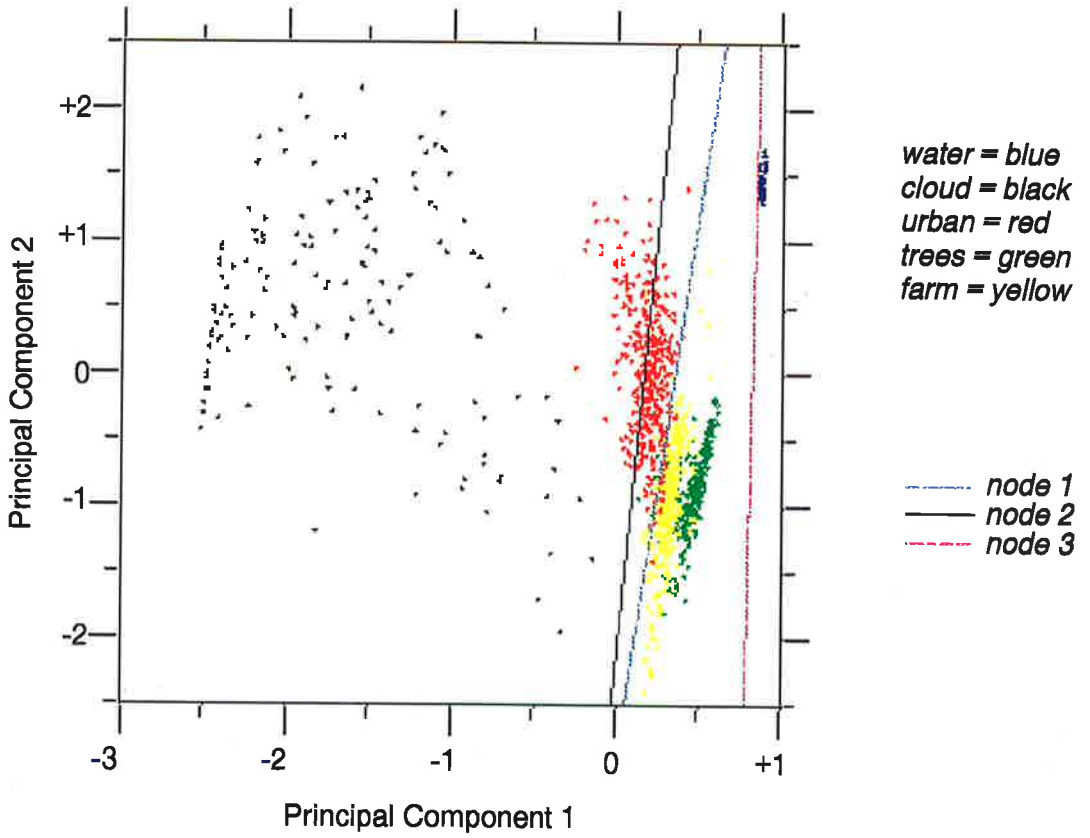


Figure 3.18b: Partitions of MSS data in PC space by MLP 4-3-5-5

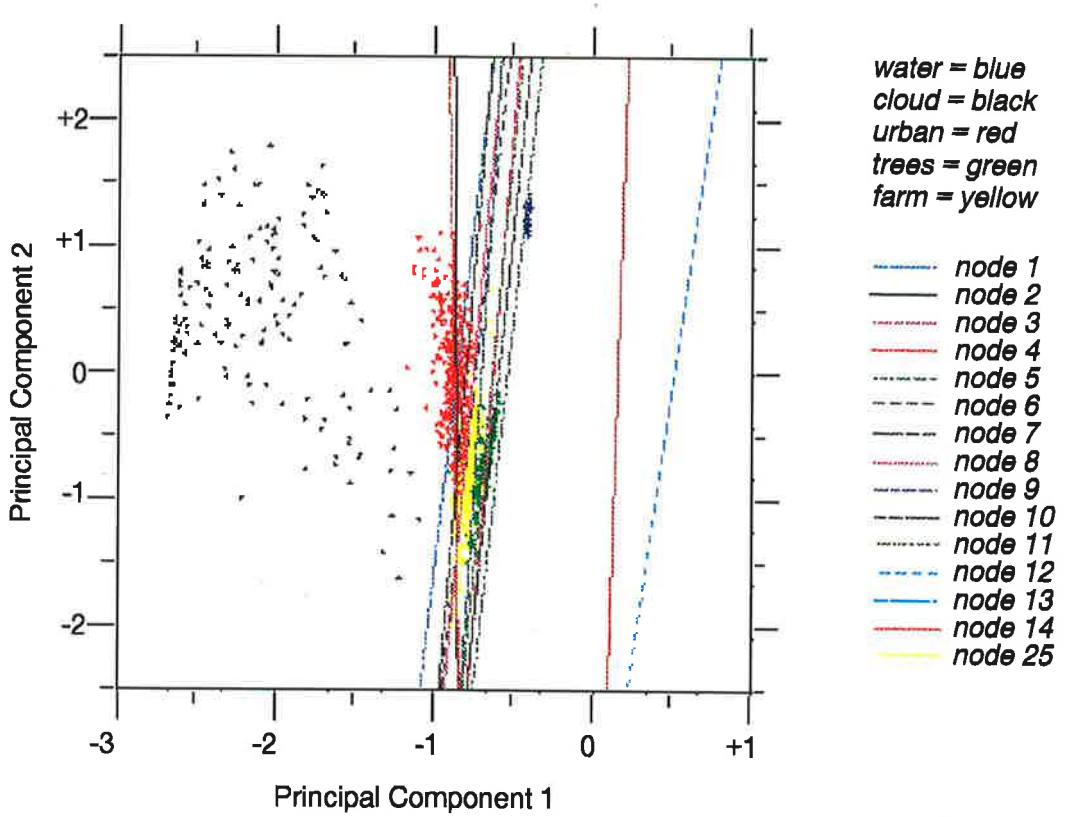


Figure 3.18c: Partitions of MSS data in PC space by MLP 4-25-6-5. 16 hyperplanes only are shown for clarity.

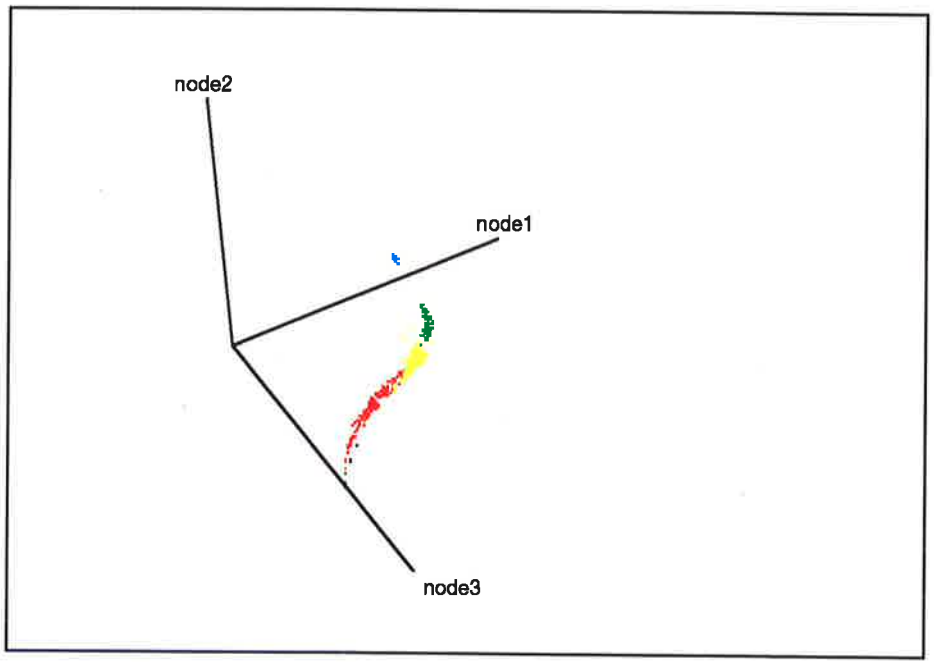


Figure 3.19: MSS data transformed by layer 1 of 4-3-5-5 MLP

3.4.4 Interpreting weights by probing with test/training data–Activation maps

A common method to assess the performance of an ANN is to map the output energy function vs the input variables. Lang and Witbrock used this method to examine the behaviour of a network designed to solve the two spirals problem [Lang88]. The spirals problem can be easily visualized using 3 dimensional plots because the output is a scalar function of two (spatial) input variables. However, the interpretation of output with respect to 4-band image data is not as simple because the output is a function of three variables, $f(x,y,band)$.

The nature of multi-spectral data is that it is sparse in spectral space, and partially correlated. Given that the best examples of likely data values are the members of the reference sets used for training (and testing), it seems logical to investigate what a network learned during training by observing the response of each node of a trained network to each element of the reference set.

To aid this investigation a new construct called an "activation map" is introduced. An activation map has the form of a set of horizontal strips, one for each node in the layer being considered (figure 3.20). The colours within a strip represent the response of the node to an input at the corresponding position in the input image. The colour key indicates the range of values. The particular colour coding was chosen to allow nodes which are inverses of each other to be easy to distinguished with the intermediate colour of white portraying an "undecided" output.

For the activation maps shown below, the input image comprises 20-pixel by 10-line subimages, each subimage containing the test set for a particular class, with classes ordered 1 to 5 from left to right. If we consider the input data to be the output of nodes in a fictitious layer 0, we can use an activation map to depict the input image used to probe the node responses. Figure 3.21 shows input data, one horizontal strip for each band of the image. The values have been normalized to lie in the (0.0,1.0) range. There are five blocks per strip, one for each class.

The activation maps for the output layer can be interpreted loosely as certainty maps. These maps comprise one horizontal strip for each output node and the 5 blocks within a strip represent that node's response to each of the 5 input blocks, one from each class. The ideal map has red blocks on the diagonal and blue everywhere else. The output layer maps shown in figures 3.22b, 3.23 and 3.24 show various degrees of mis-classification for the differently structured networks. In each of the output maps it is clear that classes 1 and 2 are well defined with particular pixels causing problems. From figure 3.21, it can be observed that the locations in class 2 where the problem pixels occur have values that are dissimilar to the rest of the training data and may be outliers or even errors in the training set. Looking at columns 3, 4 and 5 of the output layer maps, we see that classes 3, 4 and 5 are less easily separated. Comparing maps for the different structures, we see that the confidence of the classification by the 4-25-6-5 network is greater than that of the other networks. Note that the intermediate valued areas do not necessarily imply mis-classification since class assignment involves finding the max value over the 5 strips for a particular pixel position.

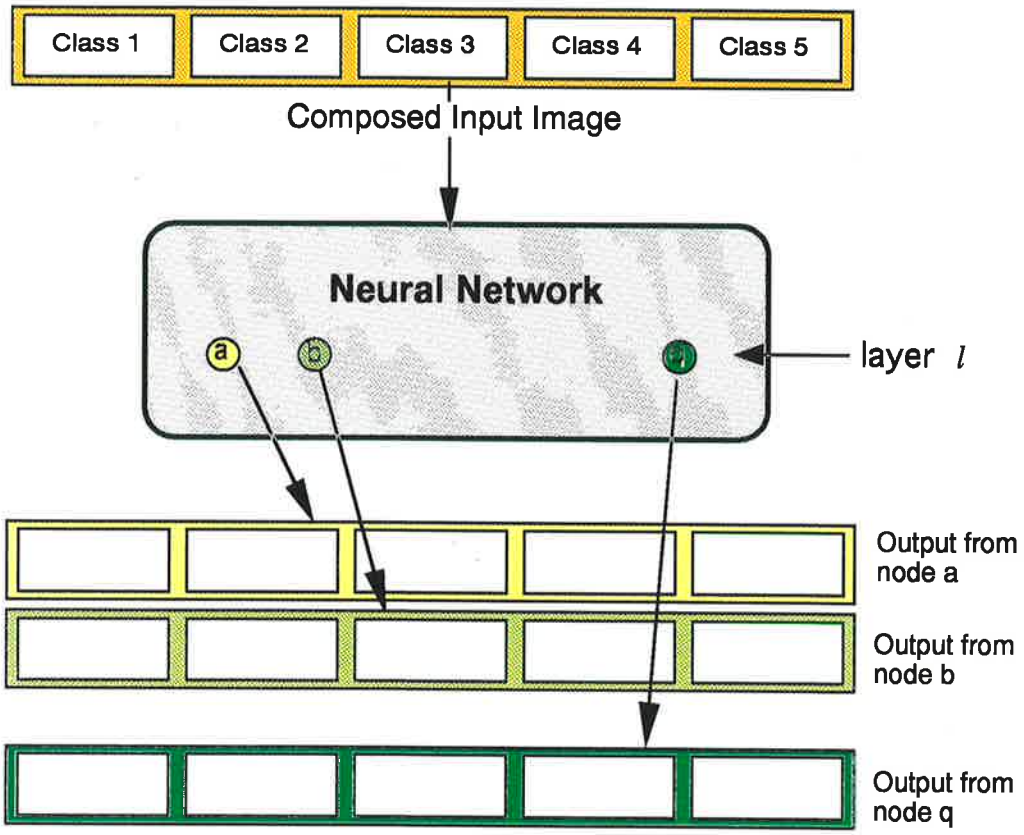


Figure 3.20: Generation of an activation map for layer l.

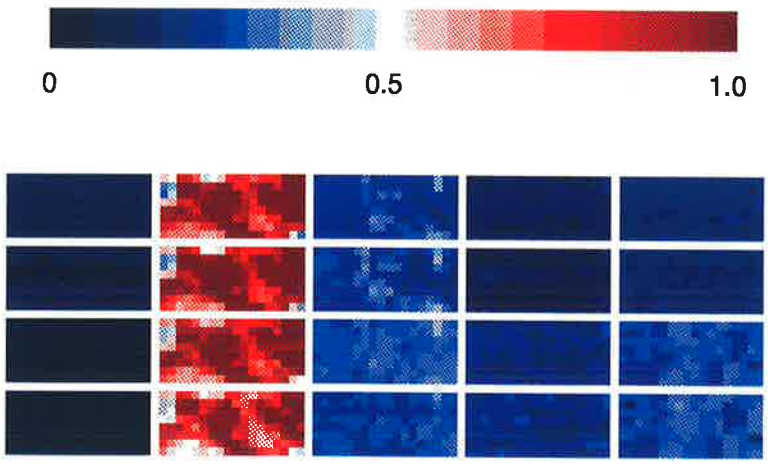


Figure 3.21: Input data as an activation map for layer 0.

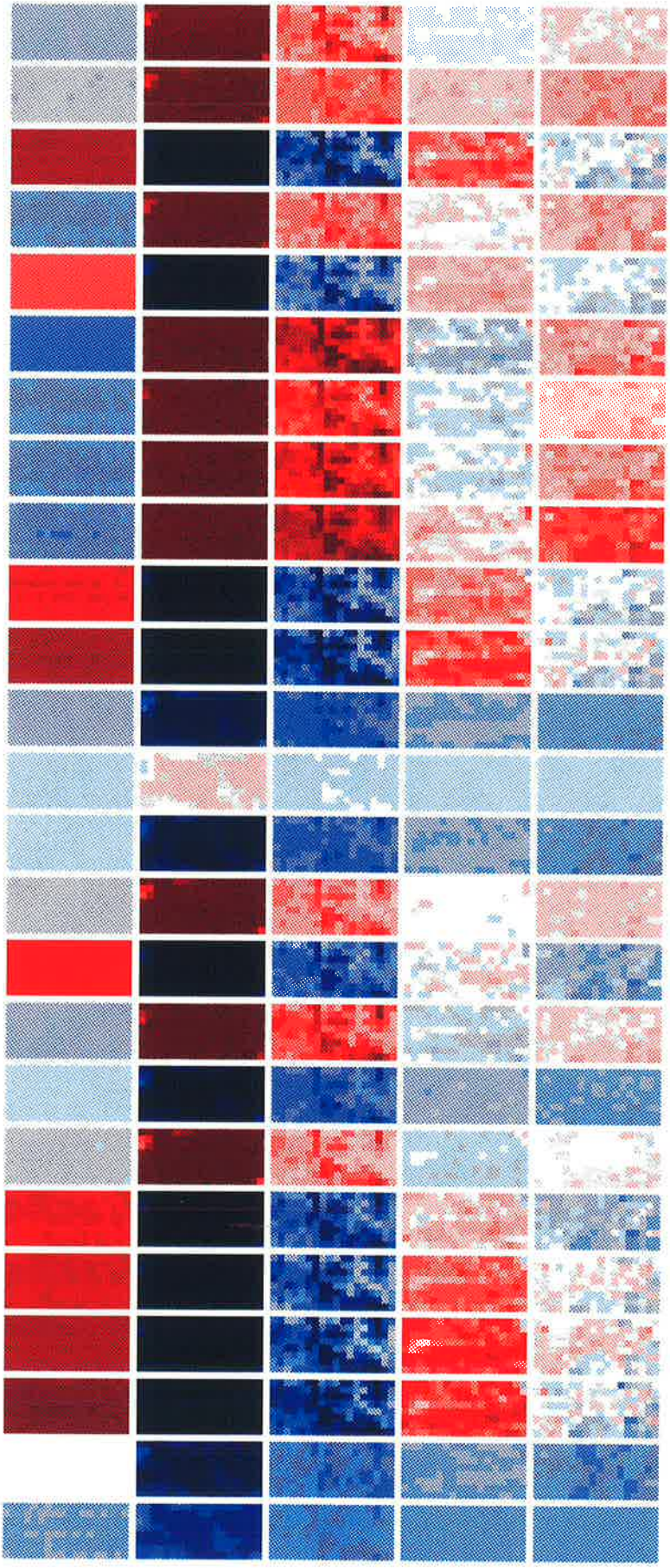
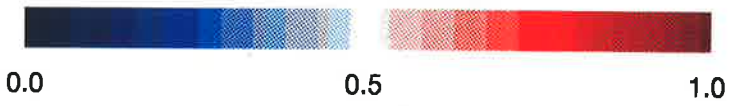
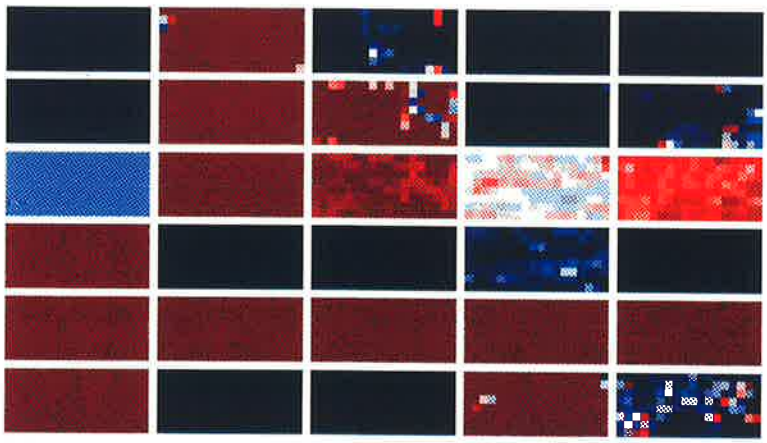
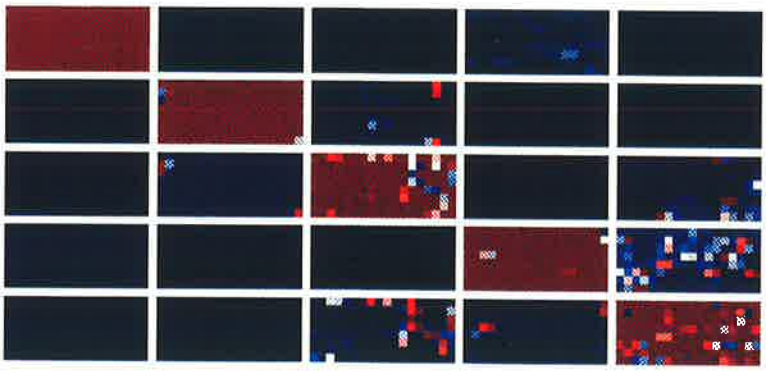


Figure 3.22a: Activation map for the 1st Hidden Layer of a 4-25-6-5 MLP





Second Hidden Layer



Output Layer

Figure 3.22b: Activation maps from MLP of structure 4-25-6-5

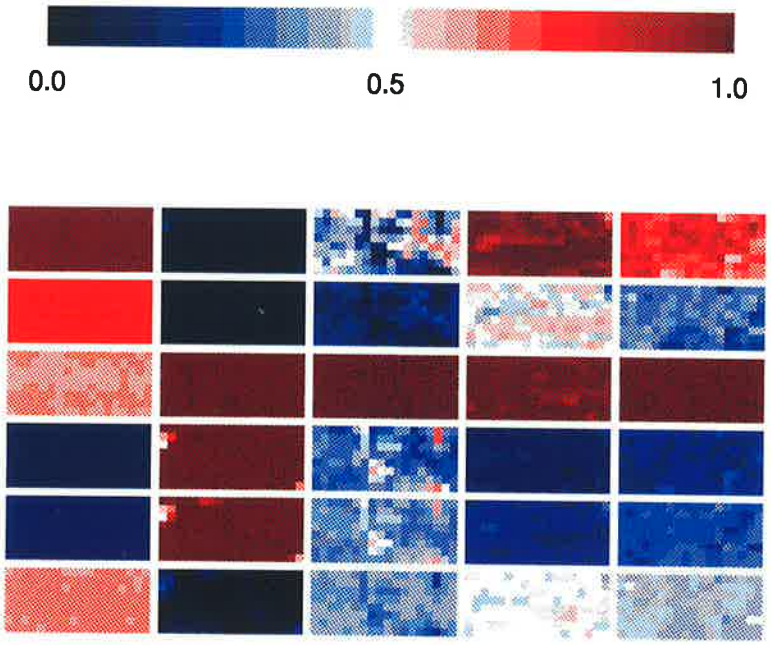
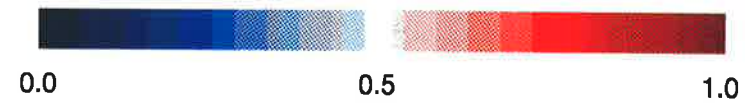
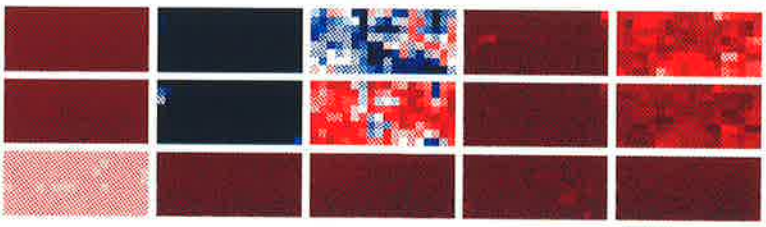
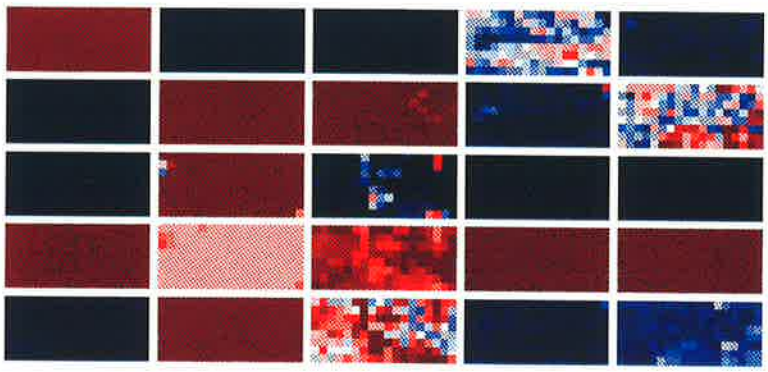


Figure 3.23: Activation maps from MLP of structure 4-6-3-5

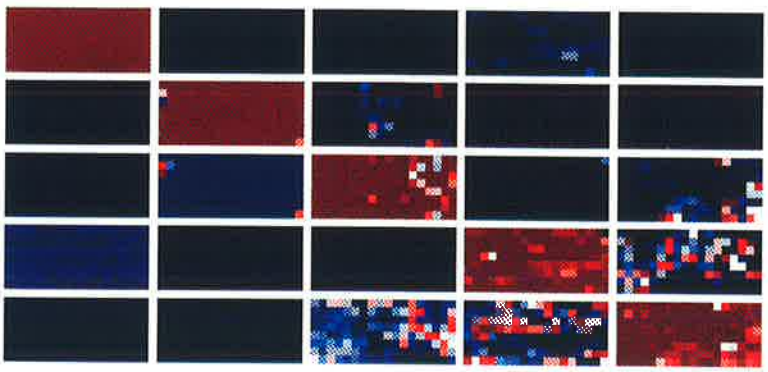




First Hidden Layer



Second Hidden Layer



Output Layer

Figure 3.24: Activation maps from MLP of structure 4-3-5-5

3.4.5 Modelling the ANN as a filter

In this section two related models which are suitable to describe the operation of the neural network are examined; in one case the ANN is considered to be acting as a set of filter banks to recognize the correct classes, and in the second case we hypothesize that the ANN performs ML classification.

The ANN is characterized as a set of filters by describing the actions of the layers in producing coded forms of the inputs. For example, for the 4-25-6-5 MLP, each node in the output layer must take 6 values from the second hidden layer and form one output indicating certainty that the current network input is from this class. In terms of the activation maps, each node in the output layer must interpret a column in the previous (2nd hidden) layer's activation map in order to deduce a class assignment. The output layer is performing a template matching with the outputs of the 2nd hidden layer.

Characterization of the function of the first hidden layer in a moderately complex network is not as easy. Clearly it needs to extract features of inputs that can be used to code for class characteristics. Examining the simplest of the activation maps for the 4-3-5-5 MLP reveals that nodes 1 and 2 develop different internal representations of class 3 but similar representations of other classes and hence are capable of extracting useful features.

The conventional classification technique, ML, produces the best results for data which has Gaussian multivariate distribution in colour space and uniform random distribution in spatial location. It is useful to analyze the activation maps to discover if a multi-layer perceptron is learning to do ML under these assumptions, since these are the only tractable conditions.

To classify each pixel, it is necessary to calculate a discriminant function from (1.11), $g_r(\boldsymbol{x}) = -\log|\boldsymbol{C}_r| - (\boldsymbol{x}-\boldsymbol{m})^t\boldsymbol{C}_r^{-1}(\boldsymbol{x}-\boldsymbol{m})$. We assign the pixel to the class for which the discriminant function is maximum. The first part of equation (1.11) has the form of a log of a class dependent quantity. This could be handled in the neural network by the bias (w_0) to the third layer since this term is a constant over all pixels that can be added after the main discriminant calculation.

The second part of (1.11) is the calculation of a Mahalanobis distance, which involves the product of the zero mean input, $\boldsymbol{x}'$ with the inverse of a class covariance matrix $\boldsymbol{C}_r^{-1}$ giving: $(\boldsymbol{x}')^t\boldsymbol{C}_r^{-1}(\boldsymbol{x}')$. The required output is a function comprising sums of terms of the form $x_i c x_j$ where x_i and x_j are elements of the input vector and c is a constant. These product terms are difficult to calculate in a linear system and can only be approximated by a finite network. It is hypothesized therefore that the first layer could form a set of weighted sums of the demeaned input vector. Taking into the account the non-linearities in each node, $x_i c x_j$ could be synthesized when the outputs of first layer nodes were summed by the second layer. The third layer could add these sums together to produce a Mahalanobis distance.

Examining the activation maps shows that this description is too simplistic. The trained network functions as a non-linear filter, but the functionality is distributed throughout the

network. Indeed, this attribute has attracted attention for its potential fault tolerance, and is one the attributes that makes and ANN attractive.

3.4.6 Redundancies

Observation of the activation maps for layer 1, indicates that some of the nodes are performing very similar tasks. For example, in the activation map for the 4-25-6-5 MLP (figure 3.22a) nodes 7 and 8 appear to be constructing the same function of the inputs. It should be also possible to detect nodes which are inverses of each other.

The visual similarity of the node responses suggests a simple automatic pruning algorithm of the form:-

For all pairs of nodes in the layer

If response of the two nodes is equal within a specified margin

for *all* members of training set (all classes)

then one of these nodes is redundant.

A more sophisticated algorithm could detect nodes which form scaled replicas of each other's responses. Removal of the redundant nodes and combining of the appropriate weights should lead to reduced calculation during classification.

3.5 Discussion

The purpose of the architecture described in this chapter is to classify multi-spectral data using the spectral information alone. This task is directly equivalent to partitioning a feature space into regions which are labelled as containing only data from one class. If the technique of classification is to be useful, it must have advantages over classical techniques that justify the relative cost of computation measured in the most general way. This is usually assessed by examining speed, accuracy and computer storage (memory) requirements. The weight given to these components of the assessment varies with currently available hardware, with currently available software, especially operating system software, and even with fashion. Some of the material included in this section is intended to try to provide the reader with some gauge of the importance given to these factors at the time of writing.

For ANNs, and for many other classification schemes, the speed of training and classification are quite different, and there are important trade-offs between speed and accuracy during the training phase. Accuracy is treated first, assuming some common trade-offs.

3.5.1 Accuracy

The accuracy of an MLP is as good as or better than the standard ML in all cases.

We expect that data taken from a multivariate normal distribution will be classified by the ML and optimal MLP classifiers with the same accuracy. It has been shown conclusively [Burrascano91], [Hampshire90], [Mackay91a] that an MLP can approximate a Bayesian classifier to any small ε, given that it is trained using the appropriate network error function.

Where there is no prior information about the probability of a particular class occurring, the most appropriate error function is the widely used function given by (3.4) [Burrascano91]. An optimal MLP is defined to have the weights giving the best possible result on test data, and implies that there are a number of sets of weights developed by performing training from different weight initializations. It remains an open question, how many trials are necessary to give a high confidence that the best MLP weight configuration discovered is near optimal.

Where data does not have an underlying probability distribution function (probability density function) conforming to the normal model, we expect that the MLP will perform better than ML because its performance is not based on a model of the data. Of course, a true maximum likelihood classifier is not limited to a multivariate normal model of data, but it would be necessary to estimate a model, possibly different for each class, and suitable statistical characteristics to allow a true maximum likelihood scheme to produce a result as good as an optimal MLP. The compact representation provided by a parameterized normal model is considered to be one of the trade-offs of the ML scheme; some accuracy is forfeited in favour of an easily collected set of parameters. As noted previously, the normal model is surprisingly robust under variations from the normal distribution.

In practice how well the MLP and ML perform is primarily a function of how well they can be trained[1]. Training in the ML (with normal model) scheme depends on how well the class's density functions can be characterized by a combination of underlying model and available training data. How well an MLP can be trained depends on the network structure, the stop-training criteria and to some extent, the random initializations of the MLPs weights and thresholds. A speed-accuracy trade-off for the MLPs used here is that training is halted after pre-determined number of weight updates, regardless of the residual error.

For the MSS reference data, selection of best MLP structure and weights over a number of trials, gave an accuracy far exceeding the MLs accuracy. Even the use of best weights for a pre-determined structure (4-25-6-5) gave an accuracy exceeding ML; an error of 5.1% vs an error 8.1%. However, training one MLP from a single initialization of the weights cannot be guaranteed to have even near-optimal performance. Further, if the network structure is inadequate, the MLP cannot learn all classes. In some cases not detailed here several MLPs became expert at classifying four classes at the expense of the fifth class.

3.5.2 Speed of classification

Speed of operation can be measured in terms of complexity of calculation and in terms of time steps. The speed of operation needs to take into account the difficulty of the calculations; for example, multiply-adds are (generally) faster than evaluation of an exponential function. On a serial machine we define a time-step to be the time for a multiply-add and assume that a functional evaluation takes significantly longer than one time-step. Add and subtract operations are assumed to take significantly less than one time-step.

---

[1] As measured by classification accuracy on an independent test set.

It is a truism that any classification algorithm "will run faster in special purpose hardware" and it is clear that both algorithms presented here for comparison will run faster on parallel hardware. However, the speed up from special purpose hardware is particularly relevant in the case of the MLP since any implementation in serial hardware can only be called a simulation. What is claimed here is that since ANNs promise to become widely used subsystems for signal processing, control and classification problems, it is likely that neural net hardware will be considered to be a general purpose "black box" that can be configured for specific problems in the same way that an array processor is utilized for specific problems. To be fair, attention is paid here to the value of special purpose hardware for ML and MLP, describing the kind of speed-ups each algorithm can expect. Also, as noted above, the speed of classification and training are separate issues. Because training of an MLP is a processes of iterative refinement of classification accuracy comparisons of classification power will be presented first.

In the simplest case of serial calculation, for the ML scheme to classify an M-band pixel into one of R classes, R evaluations of a discriminant function are required, giving $R(M^2 + M)$ multiply-add operations (2.29). The simplest parallelization is to allocate one processor to each discriminant function which reduces the number of time-steps to $(M^2 + M)$ Further parallelization could be achieved by performing some of the multiply-adds required for matrix manipulation in parallel. At the cost of $R.M$ processors, the number of time-steps could be reduced to $2.M$. With $R.M^2$ processors the number of steps could be reduced to $O(\log M)$.

For a serial machine simulation of an MLP with three active layers, classification of an M-band pixel into one of R classes requires one multiply-add for each weight plus one evaluation of the exponential function for each active node. Evaluation of the exponential in output nodes is not required during classification since class is selected using a maximum operator (which is also used in ML). Hence total operations are

$$\{ (M + 1)L_1 + (L_1 + 1)L_2 + (L_2 + 1)R \}\ \textit{mult-adds} \ + \ \{ L_1 + L_2 \}\ \textit{function-evals}$$

where the terms in the multiply-add part correspond to contribution from each active layer. Functional evaluation will take a long time unless performed using a look-up table, in which case it is assigned one time-step. Using a look-up table introduces quantization errors into the function value and the size of an appropriate look-up table is subject to memory-speed trade-off considerations [Xie91].

For a true ANN, possibly implemented in analogue VLSI [Verlysen91], the philosophy is that the operation of all nodes in one layer can take place in parallel. Hence the number of time-steps required is equal to the number of active layers (three), although a time-step will be longer on an analogue machine than a digital machine. Time taken for a classification is independent of the number of nodes and inputs. This means that there is no penalty for having a more highly dimensional input vector; for example, there will be no difference in classification time between the speed of classifying a 4-band image and a 256-band image.

On a hybrid machine, where each node is implemented as a digital VLSI processor (see for example [Watson90]), it might necessary to perform the calculation inside a node in serial, causing the number of time-steps to be

$$\{ (M + 1) + (L_1 + 1) + (L_2 + 1) \}\ mult\text{-}adds\ +\ \{ L_1 + L_2 \}\ function\text{-}evals$$

where each bracketed term in the multiply-add refers to the contribution from a layer. In general, hybrid machines need to be considered on an individual basis to gauge the degree of speed-up.

For an image with many pixels it is often advantageous to begin processing one pixel before the last has completed, a process known as pipelining. To design a pipeline processor, the classification process must be decomposed into a set of serial subprocesses. The speed of a pipeline is then controlled by the slowest subprocess. Clearly, the speed-up is a ratio of the time taken for the sum of the subprocesses to the speed of the slowest subprocess. Given that an increase in speed would be first pursued by making the process as parallel as possible, any comparison of the speed-up of ML vs the speed-up of an MLP is highly speculative because of strong dependency on the hypothetical parallel machine structures.

3.5.3 Speed of training

During training, the object is to distil the essential characteristics of the training set into a convenient, often compact, form.

In a scheme such as ML, training requires the estimation from the training data of mean and covariance for each class. This takes one pass through the input (training) data and requires $O(M.N)$ operations where N is the total number of examples in the training set[1]. Classification uses the inverse of the covariance matrices, so training generally includes another $O(M^3)$ operations to invert one covariance matrix per class.

Training an MLP requires repeated presentation of the input data, with the presentation of each sample requiring a complete classification which can take a wide range of time-steps depending on the architecture of the machine implementing the algorithm. It is sufficient to observe that in most cases training an MLP takes much longer than ML and must be justified by the possible improvements accuracy and generalizing power.

The work described in this chapter was performed on a SUN4 work-station using the LUCID image processing system. Some simulations of the neural network needed 10 minutes of training time, while the generation of statistics and the inversion of the covariance matrix took seconds. Both the neural network and ML classification processes could achieve considerable speed-up through the use of parallel processors, although in different ways. Both methods classify one pixel at a time and would benefit from the ability to handle many at once, for example, by the use of a pipeline processor. The neural network techniques used here contain scope for parallel processing of a single pixel's vector and a considerable speed-up could be expected using special purpose neural network simulator hardware that is currently becoming available.

---

[1] Assuming that N>>M, where M is the number of bands in the multi-spectral data.

UNIVERSITY OF ADELAIDE LIBRARY

3.5.4 New algorithms to speed-up training

Various researchers have attempted to reduce the number of presentations of data required to train a neural network by modifying the weight update rules used in gradient descent back-propagation training. The most often cited technique is the use of conjugate gradient techniques [Moller90]. These techniques have been developed in parallel with the work described in this thesis and have not been explored. However, it is fair to observe that the use of conjugate gradient forces weight update to occur only after a complete presentation of training data, it requires storage of the previous gradient and it need a line search in its objective function. Further, the considerable speed-up that conjugate gradient optimization affords the simulation of an MLP is not necessarily transferable to a real ANN.

Another approach has been to alter the architecture to speed training; for example, quickprop [Fahlman88], and cascade correlation [Fahlman90]. The existence of faster algorithms and various gradient descent speed-ups does not invalidate results here. The un-adorned back propagation system used in this thesis shows the capabilities of the MLP and does not exclude the use of faster weight update rules. However, some speed up techniques incorporate assumptions about probability density functions of the training data and would need to be tested with MSS data.

3.6 Summary and conclusions

This chapter has described the use of a multi-layer perceptron for the classification of multi-spectral images. Theoretically it is possible for a trained MLP to perform as well as the standard ML when data comes from an underlying probability density function that is well characterized by the multivariate normal model used by ML. Where data comes from a different distribution, it is expected that the MLP can exceed performance of the standard ML scheme.

The experiments presented in this chapter verify that for an MLP with a carefully selected structure it is possible to exceed the accuracy of ML by a considerable margin on MSS data. Further, given an over-large MLP it is still possible to routinely exceed the classification accuracy of ML, given a good selection of weights during the training phase.

As expected, the quality of training as measured by testing the trained MLP against a test set was found to be highly dependent on the randomly selected initial weights. (These represent a particular the starting point in weight space.) Repeated trials showed that when weights are trained from an arbitrary point in weight space, the resulting MLPs accuracy cannot be guaranteed to give to a good approximation to the best trained MLPs accuracy, on average.

From this it must be concluded that, to be reasonably confident that an MLP will classify accurately, weights need to be selected after repeated trials in which weights are initialized to different values, and the resulting MLP is assessed by testing its accuracy on an independent test set. To do this, the scarce resource of reference data will need to be partitioned into a training set and a test set. This is also a problem in statistical pattern recognition and has standard compromise solutions [Duda73]. The issue of how many trials are required to select a

good set of weights is beyond the scope of this thesis, but ten trials were observed to be acceptable for this kind of data

Given a trained MLP it is reasonable to ask about the internal representations of classes that are learned. This was addressed using a variety of visualization techniques, some of them novel. Partitions of input space were examined by transforming input data into a two dimensional space for ease of display and also transforming the partitioning hyperplanes defined by the first layer weights into the new space. Decision hyperplanes were observed to cross the principal component axis of the training data almost at right angles indicating best possible orientation for discrimination.

This chapter has also described the use of activation maps to probe the operation of three multi-layer perceptrons for the classification of multi-spectral images. Observations of these maps show a relationship between hidden nodes' responses and the required output suggesting that the MLP acts as a non-linear adaptive filter. It is possible to detect potential redundancies in the networks based on these observations. The maps from the output layer give an indication of the certainty of classification confirming the relative measured accuracy of the three networks. However, extracting evidence for particular internal representation within a trained network remains a difficult task.

Table 3.3a: Weights learned by MLP of structure 4-6-3-5

	Layer 3 node				
from layer 2	node 1	node 2	node 3	node 4	node 5
bias	-6.2629	-5.9569	-6.8683	3.3970	2.5731
node 1	9.6377	-5.6270	-5.7363	-5.9472	-15.1051
node 2	-5.9056	0.9108	10.7541	-16.9185	-3.3351
node 3	-3.3742	8.1276	-6.4851	-3.4880	-7.2864

	Layer 2		
from layer 1	node 1	node 2	node 3
bias	-1.2833	0.8507	-1.6925
node 1	5.5761	-8.0101	-5.7882
node 2	4.2182	-5.6047	-4.8714
node 3	-6.1265	4.1075	1.1348
node 4	-4.7323	3.3036	2.2734
node 5	-3.0455	2.2139	1.4411
node 6	0.6782	-1.0612	-2.5228

	Layer 1					
from inputs	node 1	node 2	node 3	node 4	node 5	node 6
bias	3.2690	1.5719	-1.0962	-3.4096	-2.5650	0.2563
input 1	-9.3943	-6.4258	3.7909	3.7103	2.3063	-1.4989
input 2	-9.4862	-6.7979	5.2419	3.7196	2.6605	-1.5698
input 3	-1.4269	-2.0561	4.0588	1.9346	1.4587	-0.9862
input 4	2.3908	0.2199	2.1013	0.0211	0.0910	-0.7313

Table 3.3b: Weights learned by MLP of structure 4-3-5-5

	Layer 3				
from layer 2	node 1	node 2	node 3	node 4	node 5
bias	-3.5927	-5.5364	-3.8172	2.4894	1.6955
input 1	10.1976	-6.8185	-6.4881	-6.7977	-15.9808
input 2	-6.3481	2.9967	9.9684	-16.4491	-2.6600
input 3	-4.4128	6.9919	-7.0884	-2.9171	-4.9636
input 4	-3.4015	-3.2215	-4.2100	1.8377	1.7761
input 5	-2.4595	1.0199	-0.4457	-2.4304	-3.4017

	Layer 2				
from layer 1	node 1	node 2	node 3	node 4	node 5
bias	-4.1071	4.0516	-0.4154	0.2885	1.1609
input 1	8.5755	-9.3094	-5.5768	2.3945	-3.5240
input 2	2.7576	-4.9286	-7.7911	0.4592	-3.1731
input 3	-7.6387	5.4706	2.2685	-0.3991	1.7102

	Layer 1		
from inputs	node 1	node 2	node 3
bias	4.3261	3.4979	-1.3111
input 1	-10.8343	-5.7199	4.3526
input 2	-11.8849	-5.7479	5.7598
input 3	-2.5857	-2.2470	4.2888
input 4	2.9606	-0.1531	3.0056

Table 3.3c: Weights learned by MLP of structure 4-25-6-5

	Layer 3				
from layer 2	node 1	node 2	node 3	node 4	node 5
bias	-3.2836	-2.1855	-1.0289	-1.1190	1.9446
node 1	-3.5564	7.4253	-6.8778	-2.5497	-3.6115
node 2	-4.2755	1.1047	6.5879	-5.0457	-6.3114
node 3	-2.5848	-0.8187	-0.5024	-1.1876	-0.2288
node 4	7.8865	-3.6280	-3.4332	-8.1429	-4.1314
node 5	-2.5791	-2.2815	-1.8901	-0.8851	1.1692
node 6	2.7637	-4.5155	-6.8779	6.0582	-6.2306

	Layer 2					
from layer 1	node 1	node 2	node 3	node 4	node 5	node 6
bias	-1.0663	0.8156	0.6125	-1.7423	0.9502	-1.1660
node 1	0.1950	1.8656	-0.1337	-1.9395	-0.3494	-2.8738
node 2	0.0565	0.1044	0.0301	-2.5966	0.4235	-1.3900
node 3	-3.2728	-3.9691	-0.1102	2.5318	-0.2193	4.1254
node 4	0.8638	-0.1867	0.0223	-2.5662	0.6079	-1.8355
node 5	-2.1224	-2.9054	-0.1802	0.5550	0.6972	2.6339
node 6	1.2696	4.1683	0.6468	-3.3979	-0.4025	-4.7862
node 7	0.7178	2.1122	0.3763	-2.4771	0.3346	-3.2424
node 8	0.8005	2.9064	0.0165	-3.3137	-0.0926	-3.6568
node 9	0.9529	2.3325	0.6901	-3.9188	-0.2101	-3.7592
node 10	-2.0694	-3.9679	-0.4160	1.5950	0.1007	3.6058
node 11	-2.9403	-4.9886	-0.7370	2.9134	0.5031	4.5549
node 12	-0.8559	-1.1821	-0.4420	-0.8122	-0.2542	0.1641
node 13	-0.4096	-0.1111	-0.4279	-1.0100	0.3908	-1.0953
node 14	-1.4921	-0.4048	0.1226	-0.5140	0.5377	0.2822
node 15	-0.0760	1.0901	-0.2085	-1.9139	0.4179	-2.0065
node 16	-2.5412	-1.2836	-0.0111	2.1804	-0.4603	1.5980
node 17	0.2812	2.6163	0.3959	-2.0959	0.3263	-3.0663
node 18	-0.9945	-1.4417	-0.0899	-0.2255	0.5814	0.4869
node 19	-0.1768	1.9158	0.4352	-1.7695	-0.0263	-2.1765
node 20	-2.8090	-2.2721	-0.0437	2.3529	-0.3725	2.6035
node 21	-2.4350	-4.2393	-0.1014	1.6651	0.5627	3.7659
node 22	-3.0641	-4.8343	-0.6104	2.0597	0.4137	4.6852
node 23	-3.5421	-5.2083	-0.9033	3.3707	0.1962	5.4409
node 24	-1.4611	0.4496	-0.2714	0.1478	-0.1457	-0.2591
node 25	-0.5292	-0.1136	0.3952	-0.7365	0.0965	-0.2656

	Layer 1						
from	node 1	node 2	node 3	node 4	node 5	node 6	node 7
bias	-1.3403	-0.9203	1.8538	-1.3255	0.7854	-2.1922	-1.6202
input 1	2.8394	0.5481	-5.139	1.1698	-3.7672	5.1468	2.8638
input 2	2.679	1.2904	-5.9622	1.0005	-3.9447	6.0939	3.5391
input 3	0.9834	0.9027	-2.1966	1.7054	-0.6857	1.4701	1.4392
input 4	-0.8219	0.9593	0.0781	1.2502	0.429	-0.8897	-0.2938

	Layer 1					
from	node 8	node 9	node 10	node 11	node 12	node 13
bias	-1.8007	-1.8144	1.3244	2.0202	-0.6936	-0.6626
input 1	3.8853	3.3727	-4.3215	-5.9037	-0.8161	0.4904
input 2	4.4212	4.5943	-5.6471	-6.9074	-1.2914	-0.1107
input 3	1.2038	1.9838	-1.8871	-2.1179	-0.2136	0.2559
input 4	-0.1558	0.4281	1.0527	0.6591	0.4608	-0.3322

	Layer 1					
from	node 14	node 15	node 16	node 17	node 18	node 19
bias	-0.4769	-1.0118	0.7998	-1.4275	-0.3692	-1.0828
input 1	-0.6971	1.4481	-2.4005	3.1526	-0.9597	2.1245
input 2	-1.0592	2.0125	-2.2143	3.4688	-1.9821	2.4187
input 3	-0.282	0.4478	-1.8502	0.7973	-0.7396	0.1861
input 4	-0.572	-0.083	-1.353	-0.9524	0.4886	-0.7169

	Layer 1					
from	node 20	node 21	node 22	node 23	node 24	node 25
bias	1.1801	1.5164	1.9519	2.3704	-0.4616	-0.8399
input 1	-3.3507	-5.0601	-6.0999	-6.5423	-0.0494	-0.3893
input 2	-3.8044	-5.5509	-6.7766	-7.7671	0.0239	-0.061
input 3	-1.8242	-1.8575	-1.6319	-2.481	-1.1977	-0.3317
input 4	-0.9062	1.0572	0.9334	0.5966	-0.9323	0.0981

Chapter 4

ANNs for Incorporation of Spatial Data in Classification of Multi-Spectral Images

Summary: Ideally, an artificial neural network should be able to discover the concept of texture without aid if it is trained with all data from a neighbourhood. An MLP using 3x3 neighbourhood information gives an improvement in classification accuracy over the simple MLP approach, but at the cost of much greater network complexity. A more efficient scheme is proposed that uses spatially structured but otherwise un-preprocessed data as input to the neural network. By augmenting a pixel's multi-spectral data with data from particular shapes ("cliques") from within the pixel's neighbourhood it is possible to significantly reduce the complexity of the network and achieve a marginal increase in classification accuracy for some data. There are many other possible extensions of the image-classifying multi-layer perceptron that allow incorporation of texture-like information. Texture information can be incorporated explicitly by augmenting input vectors with measures from conventional, predefined texture feature generators. For comparison, some experiments were carried out to test this approach, but it is computationally expensive, makes assumptions about the nature of texture and needs large neighbourhoods, reducing apparent image resolution.

4.1 Overview

The aim of the work described in this chapter is to make use of spatial information in an image to improve classification accuracy obtained with an MLP. An MLP with adequate capacity and an unlimited number of examples from which to learn could ideally learn about the spatial structure of the multi-band radiance values, taken over a neighbourhood of the pixel to be classified.

The expectation that an MLP could perform this function under ideal conditions is based on the study by Minsky and Papert of the "perceptron" as a machine that could learn to recognize geometry of structures in binary images [Minsky69]. Minsky and Papert showed that there were certain structures that a single layer perceptron could not learn under any circumstances.

They briefly discussed multi-layer networks for general structure recognition, but did not go further because at the time there was no training method available. With the publication of the back propagation algorithm, this was no longer true.

In this chapter, it is asserted that spatial constructs, called "texture" by image interpreters, are repeated micro-structures which could be recognized (under ideal conditions) by a multi-layer perceptron. Regularly repeated microstructures form the basis of texture in the work of Lu and Fu [Lu78], Vilnrotter [Vilnrotter86] and to a lesser extent Rao[Rao90]. What is conjectured here, is that to be recognized consistently by an MLP, it will not be necessary for the structures to be repeated in a deterministic sense, but only in a stochastic sense.

Further, it will also be central to the theme of the chapter (as indeed it is to this thesis) that colour texture (multi-spectral texture) is a image construct that carries more information than the sum of individual monochromatic textures. Therefore, we will treat neighbourhoods as arrangements of coloured entities and will not model the image as M separate bands which each give rise to separate texture features (an approach used by Kamata *et al.* [Kamata91]). Any decomposing of the input information will be on a spatial rather than spectral basis.

Other researchers have included spatial information in multi-spectral image-classifying ANNs. Eklundh *et al.* used some neighbourhood information in an unsupervised MSS-classifying associative net [Eklundh86]. Hepner mentioned the possibility of including spatial data but gave no details [Hepner90]. For monochromatic image-classifying ANNs it is common to use a preprocessor to give a multidimensional input vector with components corresponding to texture features; there are examples of the use of first order textures features as input to an MLP [Blanz90] [Gish89] [Gish90], Haralick's texture features [Haralick73] with a linear vector quantizer ANN [Visa90], Laws' texture features [Laws80] with a probabilistic relaxation-based ANN [Hsiao90] and Zernike moments with probabilistic relaxation [Khotanzad90].

The aim here was to use the MLP to learn appropriate preprocessing by supplying it with enough neighbourhood information for each pixel (and enough training pixels). Section 2 of this chapter presents some examples of classifications of multi-spectral images based on a 3x3 neighbourhood. The input to the MLP comprised all the multi-spectral data from a 3x3 window centred on the pixel to be classified. The classification methods were tested using data from Landsat MSS, Landsat TM and SPOT-XS (multi-spectral) images. Improvement in classification accuracy over the spectral-only MLP was observed for each data set tested.

To increase accuracy further, the input of the MLP was structured to provide "hints" to the MLP about the nature of spatial adjacency. In this approach, claimed to be unique, individual nodes in the first active layer were assigned information from particular "cliques" in the input window. In some cases, this improved accuracy. In all cases this speeded training and classification using a simulated MLP. In an MLP realized in hardware this approach would represent an important saving in complexity.

For comparison, some texture feature generators, including PROC from Chapter 2, were tested as preprocessors for use with an MLP. Some of the preprocessors tested gave good

results but at the cost of imposing *a priori* models on the nature of expected texture. Practical problems arise in implementation, with some methods being extremely time consuming. Results for full image classification did not appear to justify the amount of computation required for the preprocessing operation.

In all approaches tested, the use of the MLP as the classification engine gave better accuracy than the use of (equal priors) ML. It is clear that the MLP is able to develop a better internal model of the (possibly pre-processed) data distribution than the assumed multivariate Gaussian distribution used by ML, and for which ML is optimal.

4.2 An MLP using neighbourhood data

4.2.1 Learning texture in an unstructured neighbourhood

With an appropriate set of weights on connections from inputs to the first active layer, all of the data in the neighbourhood of the pixel to be classified could be a source of spatial information. For example, a class called "suburban" might consist of semi-regular blocks of green lawn, red roof and blue swimming pool. Another class called "recreation parks" may also contain green lawn pixels but not in the same spatial relationship. We would like to be able to distinguish between the two information classes, and are not interested in swimming pools or lawn as such. The aim is to use the MLP to learn texture as micro-structure that might be characteristic of a particular class by observing examples consisting of a pixel's pattern vector, the desired class label, and the pattern vectors of all the pixels in the neighbourhood of the pixel to be classified.

Given that we accept the existence of textured classes, then it is necessary to amend assumption A1 of Chapter 1 so that the measurements comprising the image X are no longer constrained to be completely local. Assumption A1 is a consequence of the requirement that

$$P(x_{kl}|X\backslash x_{kl}, T) = P(x_{kl}|T) \tag{4.1}$$

where $\backslash x_{kl}$ means *excluding* x_{kl}. Allowing the existence of texture, the model becomes:

$$P(x_{kl}|X\backslash x_{kl}, T) = P(x_{kl}|X_{kl}^{\eta}\backslash x_{kl}, T) \tag{4.2}$$

where X_{kl}^{η} are the data in a neighbourhood of (k,l), and the neighbourhood is defined as before by $\eta_{kl} = \{(i_1,j_1), (i_2,j_2),\ldots (i_{N_\eta-1},j_{N_\eta-1}), (k,l) : (i_n,j_n) \subset L\}$ such that $(k,l) \in \eta_{ij} \Leftrightarrow (i,j) \in \eta_{kl}$. This is similar to the condition required to model the image as a Markov random field (MRF) model [Derin89].

The model used here differs from an MRF in two aspects. In the conventional model, it is assumed that it is possible to measure a "texture" function that characterizes spatial structure within the whole image, by examining neighbourhoods. Here we only want to characterize spatial structure within data from a particular class, and we imagine that different classes will have different textures. Also, when using a strict MRF model, it is necessary that neighbourhoods are non-overlapping. Here we relax that requirement.

The process of classification involves scanning the image with a moving window that identifies the neighbourhood data to be provided to the MLP to classify each pixel. By

choosing the moving window from the class of ordered hierarchical windows defined in Chapter 2 after [Derin89], the size and shape are determined by a single "order" parameter[1] (examples given in figure 4.1). This system of windows has some symmetry properties which will assist the MLP to be independent of certain texture micro-structure reflections and rotations. The use of these ordered windows also imposes the requirement of spatial adjacency on the pixels in a neighbourhood making the concept of neighbourhood more intuitive.

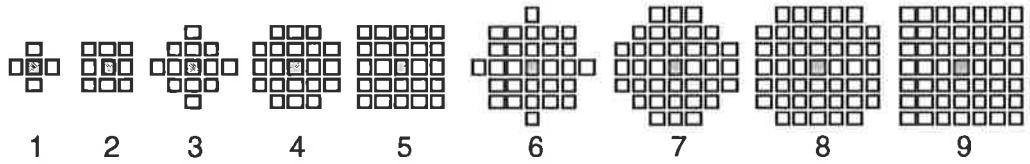


Figure 4.1: Hierarchical neighbourhoods with orders 1 to 9.

The scale of the texture is important in determining the size of an appropriate moving window. Some experiments carried out on monochromatic textures reinforced this view. Standard monochromatic textures [Brodatz66] were formed into a test image and classification was attempted using a 9 input MLP (a 3x3 neighbourhood). Normalized, 256x256 images of D24 and D112 (Brodatz's numbering) were split into left and right halves (RH and LH). Networks with various order inputs were trained on D24 LH and D112 RH, and tested on D24 RH and D112 LH (figure 4.2 shows a sample of the test image).

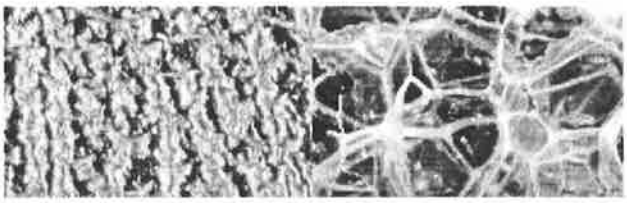


Figure 4.2: Monochromatic texture test example.

It was not possible to reduce the mean square error (mse) of any of the MLPs tested below approximately 0.2 for 100,000 exemplars[2]. Results of classifying the test set with final weights (i.e., after 100,000 updates) indicated that an MLP with information from a large neighbourhood would be required to give reasonable classification accuracy (figure 4.3). Examination of the texture example reveals that the distance between features is quite large compared to the size of the lower order neighbourhoods, explaining the problems in learning any characterizing features.

---

[1] The order parameter is related to the radius of the symmetrical neighbourhood. Order zero is the pixel itself. Each new order increases the radius by enough to include the mimimum number of extra pixels, while maintaining symmetry.

[2] As in Chapter 3, a weight update was made after each exemplar presentation.

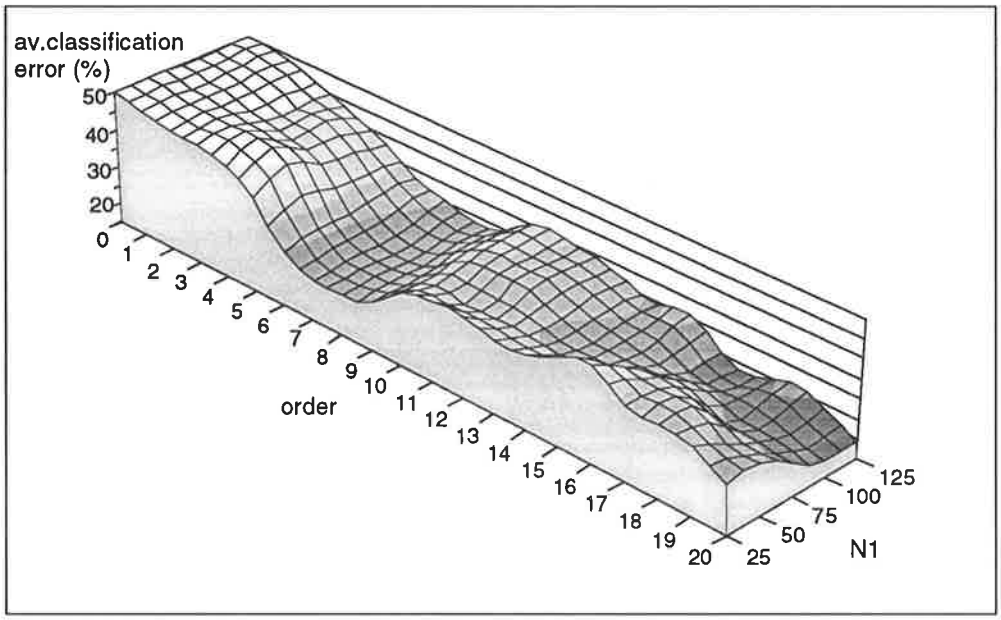


Figure 4.3: Error vs order for various 9-N1-6-2 MLPs on the monochromatic texture discrimination problem.

In the computer vision literature, texture features have been generated from neighbourhood sizes ranging from 3x3 [Unser89] to 11x11 [Laws80] and beyond. While a larger window is desirable to capture long range correlation in the image and hence large features, the larger the window, the more difficult it is to define edges of the textures and there is a consequent loss of resolution in the output image. The converse problem also exits; given that a texture can only be characterized by a microstructure of size at least A, then it would be incorrect to label any group of pixels less than size A as having that texture.

The problem of scale can be resolved by using domain specific parameters. The domain considered here is restricted to images from satellite sensors which are appropriate for wide area classification. Initially, a 3x3 (second order) window was investigated since we wish to retain the highest resolution possible and also to avoid excessive calculation. The use of a first order neighbourhood was rejected because it would not allow diagonal pixel relationships. The 3x3 window limits the sort of texture that can be recognized to that in which variation is short range with varying pattern vectors occurring in semi-regular groups of 9 or fewer pixels. The kind of land cover to which this translates is governed primarily by the resolution of the satellite sensor. For Landsat MSS with 80m resolution, classes with (physically) large scale features, such as "cultivated lands", could benefit from 3x3 windows. For SPOT-XS (multi-spectral) with 20m resolution and Landsat TM with 30m resolution, the benefits will apply to other classes with smaller scale features, for example, "urban". The higher resolution satellite images are more likely to have classes with texture since the improved resolution will lead to the detection of short range variation that was previously hidden by the filtering effect of the lower resolution sensors and associated optics.

A simple preliminary test to check that an MLP with a 3x3 window can learn microstructure was carried out using an image in which one half was a checkerboard and the other half was

Gaussian distributed noise with the same grey-level distribution function as the checkerboard (figure 4.4). The hypothesis was that if an MLP with a 3x3 window could learn structure then it would be able to differentiate between the two halves of the image, one with structure and one without, whereas an MLP that could only see a single pixel at a time would not be trainable. This hypothesis was confirmed using MLPs of structure 1-25-2 and 9-25-2. An MLP with first order neighbourhood was also able to learn to distinguish the two halves of the image.

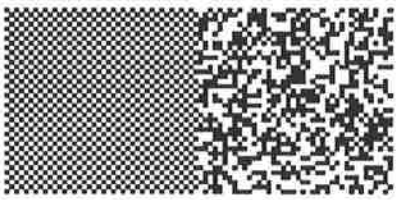


Figure 4.4: Image for a simple microstructure discrimination test.

Each half of the image presented in figure 4.4 has equal numbers of pixels selected with 2 possible grey levels, giving an exactly matched grey-level probability distribution functions.

4.2.2 Neural network implementation

The architecture of the MLP used previously to classify multi-spectral images was modified to take its inputs from a 3x3 window. For generality of function an MLP with 3 active layers was retained. The 3x3 window fixes $N_0 = 9M$ inputs (M spectral bands) and the number of outputs is also fixed with one output for each class giving $N_3 = 5$ for all problems presented here. N_1 and N_2 are selected to be trainable, rather than optimal. $N_1 = 25$, 50, 75, 100 and 125 were tested initially with $N_1 = 25$ showing marginally better results, and considerably faster speed in simulation. Variation of the number of nodes in the second active layer was also tested. $N_2 = 4$, 6 and 8 were tested, with $N_2 = 4$ found to cause classes to be eliminated $N_2 = 8$ was found to give similar results to $N_2 = 6$, but used more time to train. Learning rate and momentum were selected as before. The structure adopted for testing performance on various data sets was $9M$-25-6-5 and is referred to as the windowed-input MLP (WMLP). Figure 4.5 shows the WMLP where $M = 4$.

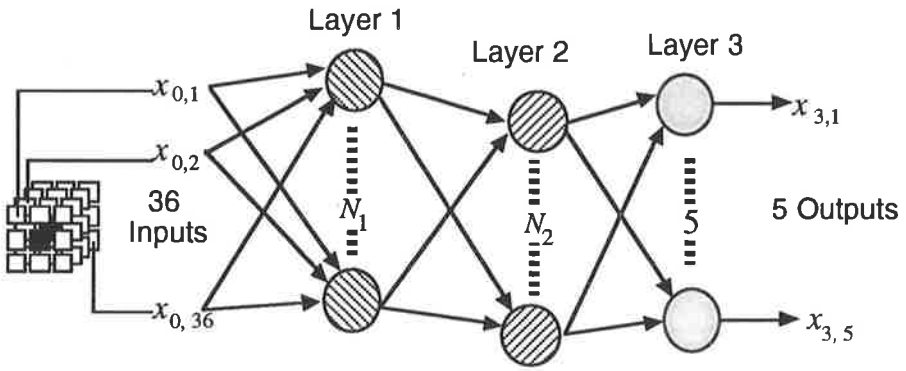


Figure 4.5: 2nd-order-input MLP classifier for a 4-band image.

During training, examples were taken at random from a particular class's training set, with the particular class also selected at random. This stratified sampling regime was followed to

remove any bias caused by different sized training sets. This should also partially compensate for the use of overlapping neighbourhoods since it makes the probability of successive weight updates coming from correlated neighbourhoods ≈ (size of neighbourhood)/(size of training set). Any rotational invariance must be acquired by the network seeing rotated examples of different microstructures during the training phase.

4.2.3 Experiments

Data from Landsat MSS ("MSS"), Landsat TM ("TM") and SPOT-XS sensors were available for this test. Each image was classified by a human operator to give reference areas typical of 5 classes. Reference areas were divided (arbitrarily) into test area and training areas. Each classifier was trained on the training areas only and tested on the test areas. This procedure is used because an MLP classifier can score highly when classifying its own training set if it simply learns each point in the input data set. To score well in a cross validated test, it must develop a suitable internal model of the training data to successfully classify unseen test data. For all data, classifications made using ML and the spectral-only MLP are presented for comparison.

In the light of the conclusions of Chapter 3, each MLP structure was trained with each set of data from 10 different weight initializations. Tables at the end of this chapter contain the accuracy measured for individual trials, while the results for the best trials are summarized and presented within the body of the text in table 4.1.

The same MSS test and training sets used in previous chapters were used here to test the MLPs with spatial inputs. The five classes loosely represented "water", "cloud", "urban", "trees" and "farm". Since areas immediately adjacent to the training and test sets were not labelled, the classifying schemes using second order neighbourhoods were not able to classify a border of one pixel's width around each of the training and test areas. For fairness of comparison, ML and spectral-only MLP classifications were re-assessed using only the data within the borders. Their adjusted accuracies appear in table 4.1.

A TM image of approximately the same area as the primary MSS scene was obtained and reference areas were selected to be in approximately the same place as the MSS reference areas. There is no cloud in the TM scene, so to give a five class comparison, areas in a quarry were selected as the substitute class. These areas on the ground are mostly of exposed sand, giving high reflectance in all bands. The resulting classifications also correctly identified the beach sand on the sea shore.

Because the resolution of the TM image is 30m compared with 80m of the MSS, more training and test examples were generated from the test areas, and the one-pixel-width borders correspond to less area on the ground.

A SPOT-XS image of the same area was not available, and similar area to the north of the MSS scene was used. The SPOT-XS resolution of 20m gives a lot of data for an area the size of the MSS image used above. For convenience it was necessary to use a SPOT image of approximately one quarter of the geographical coverage of the MSS and TM images. Ground

truth provided by a field party for another purpose identified 52 areas coming from 5 information classes corresponding to "roofs", "pavement", "ovals" (heavily watered recreational greens), "trees", and "grassland". This set of information classes was chosen to be realistic but challenging for the classifiers. Some classes proved inseparable and gave large classification errors. Partly because of the resolution available and partly because of the method of collecting ground truth, which was specifically intended for ML classification, individual areas tended to have little spatial structure. Inclusion of multiple areas with different spectral characteristics in a single class produced multi-modal classes in some cases. Areas from a particular class were arbitrarily separated into test area and training areas for using in accuracy assessment as in other experiments.

The accuracy of the WMLP classification exceeded the accuracy of the spectral-only MLP for all data sets (a summary in table 4.1, and detailed results in tables 4.2-4.4). In the case of MSS and TM data the improvement was substantial. For the SPOT-XS data, there was a substantial increase in classification accuracy from ML to MLP, probably due to a multi-modal class, but marginal increase in accuracy from the use of MLP to WMLP. As observed above, the individual reference sets lacked spatial structure with the consequence that neighbourhood pixel values contributed little spatial information.

Examples of overall image classification are provided in figures 4.9–4.11. The use of a moving window produces an inevitable smoothing indicated by the (desirable) removal of isolated misclassified pixels, and the (undesirable) blurring of transitions between classes. Smoothing problems are also common with neighbourhood-based texture pre-processors, and are addressed in part, by Chapter 5 on context. In the MSS image, the area of cloud shadow (figure 4.9), which is unlike any of the training examples, is not classified well. This could indicate that the classification is mean-value based but it is more likely that useful spatial detail has been lost due to quantization effect on low values from the sensors.

4.3 MLP with a structured neighbourhood.

4.3.1 Structuring an MLP to facilitate texture learning

Using a moving window corresponding to a second order neighbourhood imposes the concept of spatial adjacency on the classifying network, but only in a mildly structured way. A more structured neighbourhood system can be defined using micro-structures known as "cliques". Cliques are combinations of pixels, all of which are neighbours (figure 4.6). They have been used successfully in image restoration (see [Derin89] for refs.).

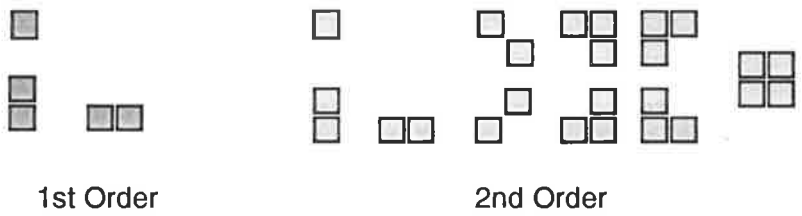


Figure 4.6: First and second order cliques.'

Cliques are used to impose a structure on the spatial information at the input of the MLP. The connections to each node in the first active layer are restricted to use only data from pixels in a specific clique. This approach follows that of Hinton [Hinton87], who configured his neural network for classifying one dimensional patterns so that the concept of spatial adjacency was built into the structure. It is emphasised again that the structuring of the input data is on the basis of spatial structuring and not spectral segregation. Each pixel to be classified and each of the pixels comprising its neighbourhood is treated as a multi-spectral entity.

The neural network described below is structured to use the 25 cliques of second order that include the pixel of interest (figure 4.7). Higher order cliques may be useful but the number of cliques explodes with order. Some inconclusive tests were conducted with MLPs using 64 cliques of third order and 124 cliques of forth order but these took an inordinate amount of time to simulate and were considered impractical. With suitable hardware this would not be the case.

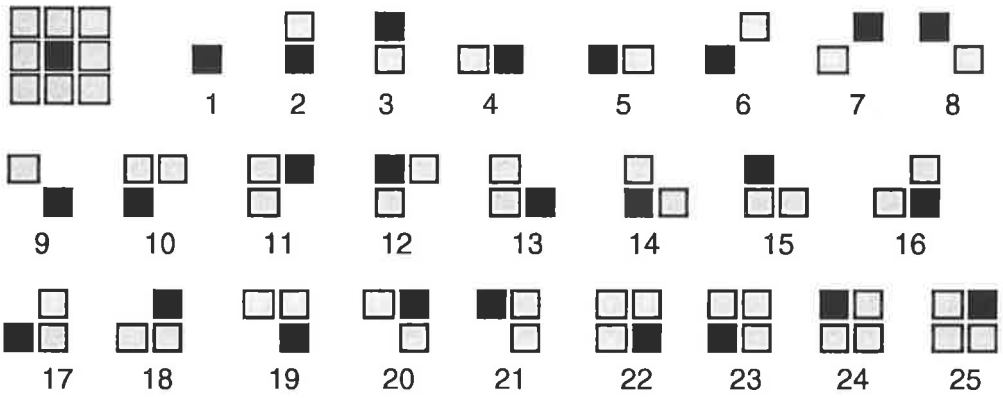


Figure 4.7: The 25 cliques of second order.

With clique structuring it is possible for a particular node (corresponding to the appropriately shaped clique) to become a "house-with-swimming-pool-and-lawn" detector. As with WMLP, rotational invariance must be learnt from suitable examples in the training set. Of course, it is possible for the WMLP to develop the same internal "clique" function for layer 1 nodes and this pre-structuring is meant to act as a *hint* to the network. A biological analogy is clearly that certain neurons in the brain are already configured for particular purposes before neural pathways form, so that certain areas of the brain have the same function from individual to individual even though the developed neurons in those areas may have different pathways expressed[1].

4.3.2 Neural network implementation

Implementation of this scheme is by setting some weights from layer 0 to layer 1 nodes to zero and not updating them during training. An example of the architecture of an MLP to classify 4-band images into 5 classes is given in figure 4.8.

---

[1] Of course, no claim is made here that an MLP with clique structured input has any biological analogue.

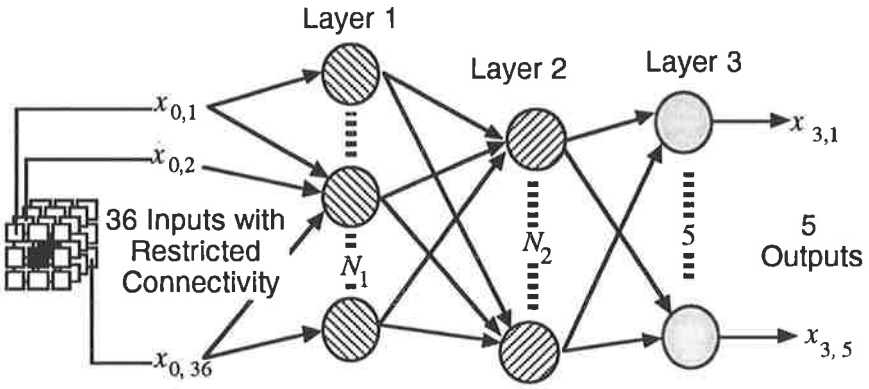


Figure 4.8: MLP classifier for a 4-band image with inputs augmented by specific cliques.

To have at least one node in the first active layer assigned to each clique it is necessary to have at least 25 nodes in that layer. Initially an MLP of structure 36-125-6-5 with 5 nodes for each clique was tested with MSS data. With nodes using Heaviside activation functions, this structure would allow each clique enough nodes to form surfaces to enclose a region in 4-D input space. N_1 values of 25, 50, 75 and 100 were also tested. Very little difference was found between values and 25 was chosen to allow faster simulation. Alternative values of N_2 were also tested with values 4 and 8 giving the same kind of results as for the WMLP.

To try to make comparisons both between data sets and between classification methods fair, all the experiments presented here use the same basic $9M$-25-6-5 structure as WMLP but with the interconnections between inputs and first active layer nodes restricted to cliques. This architecture is referred to as clique-structured-input MLP (CMLP).

4.3.3 Experimental results

The results of classifying the MSS data sets using this method were encouraging (table 4.1). Average classification error fell marginally from 0.3% without cliques to 0.2% with cliques and the maximum error made in classifying any individual class fell from 1.5% to 0.5%. Viewing the whole image classification (figure 4.9), it can be observed that both MLP classifiers that use neighbourhoods smooth out isolated pixels in the output image, but that the CMLP is still able to preserve some edges.

Overall, the accuracy measured on test sets represents represent a considerable increase in accuracy over the ML classification (with 6% average error) without the need to provide any extra training examples. However, given that these results are the best of ten trials and the error is already quite low, it would be unwise to select a CMLP over a WMLP on the basis of accuracy alone.

Using CMLP, classification accuracy of TM data did not improve over the WMLP. Error increased from 2.0% to 3.8%, but was still significantly better than the spectral-only ML and MLP. It should also be noted that while the best result over ten trials showed the WMLP gave better accuracy, the results on average were better for the CMLP (see table 4.3).

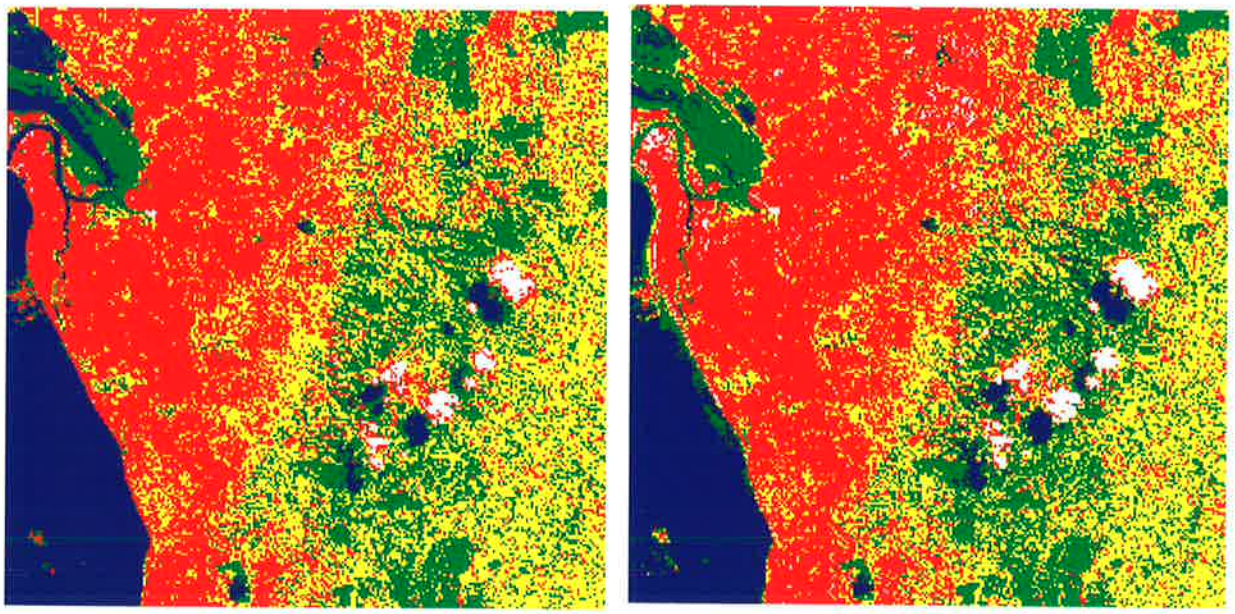
Classification of SPOT data by the CMLP showed no improvement over the WMLP in average classification error or maximum classification error for any one class.

While the gains in accuracy using the CMLP are at best marginal, the speed up in training time and classification time for a simulated CMLP compared to a WMLP are marked because of a reduction in the number of adjustable weights on each node in the first active layer from 9 to an average of 2.7. During training this improves the speed of the forward classification pass, the backward error propagation pass and the weight adjustment pass.

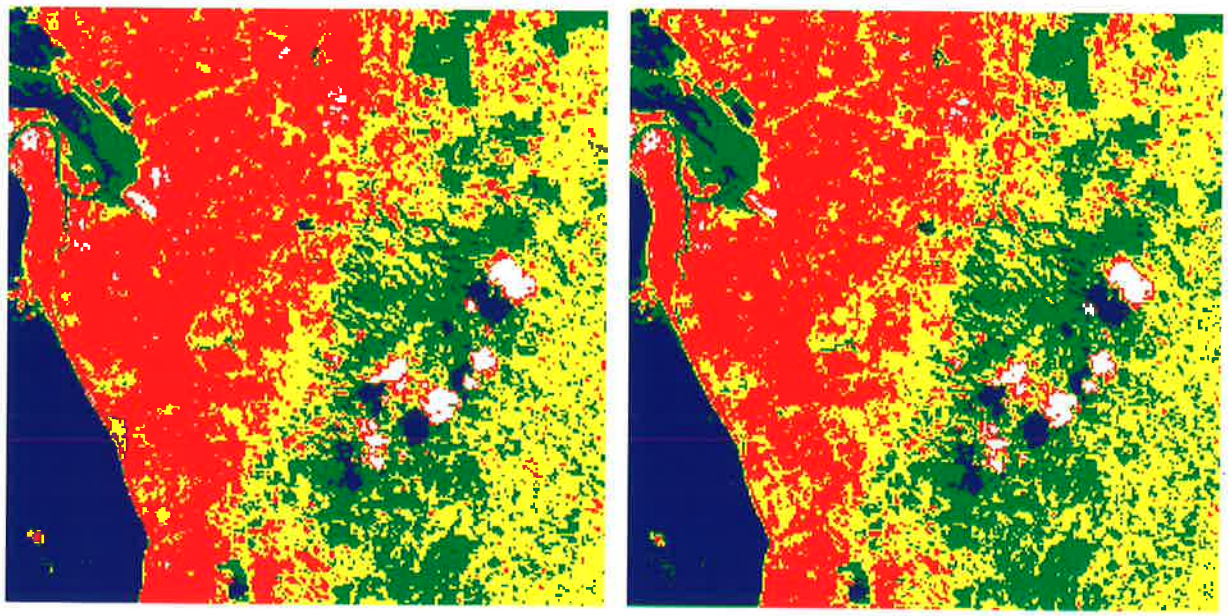
In a hardware VLSI implementation of an ANN, the restriction of interconnections from inputs to first layer nodes translates into a reduction in the need for interconnecting tracks on the VLSI die, which is a goal strived for in much ANN VLSI research (for example see [Redding91]).

Table 4.1: Summary of classification accuracy for spatial-MLP experiments

Classifier	Data	Max error in any class	Average error
ML	MSS	15.5%	6.1%
MLP	MSS	10.5%	4.2%
WMLP (3x3 moving window)	MSS	1.5%	0.3%
CMLP (clique structured inputs)	MSS	0.5%	0.2%
ML	TM	12.4%	7.1%
MLP	TM	11.1%	4.6%
WMLP (3x3 moving window)	TM	4.1%	2.0%
CMLP (clique structured inputs)	TM	9.3%	3.8%
ML	SPOT	49.3%	21.7%
MLP	SPOT	33.3%	14.6%
WMLP (3x3 moving window)	SPOT	37.9%	13.5%
CMLP (clique structured inputs)	SPOT	37.9%	13.5%

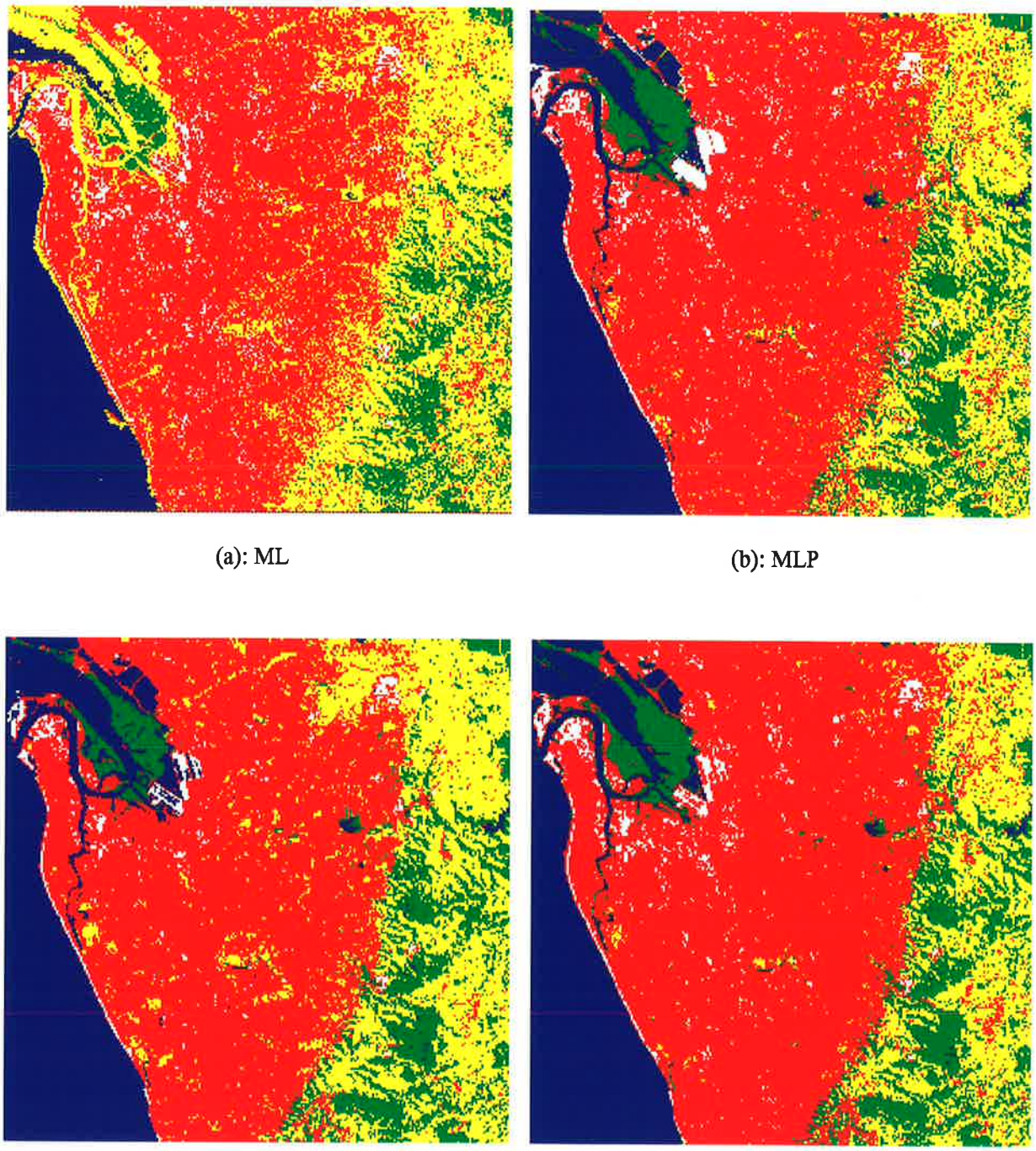


(a): ML (b): MLP



(c): WMLP (d): CMLP

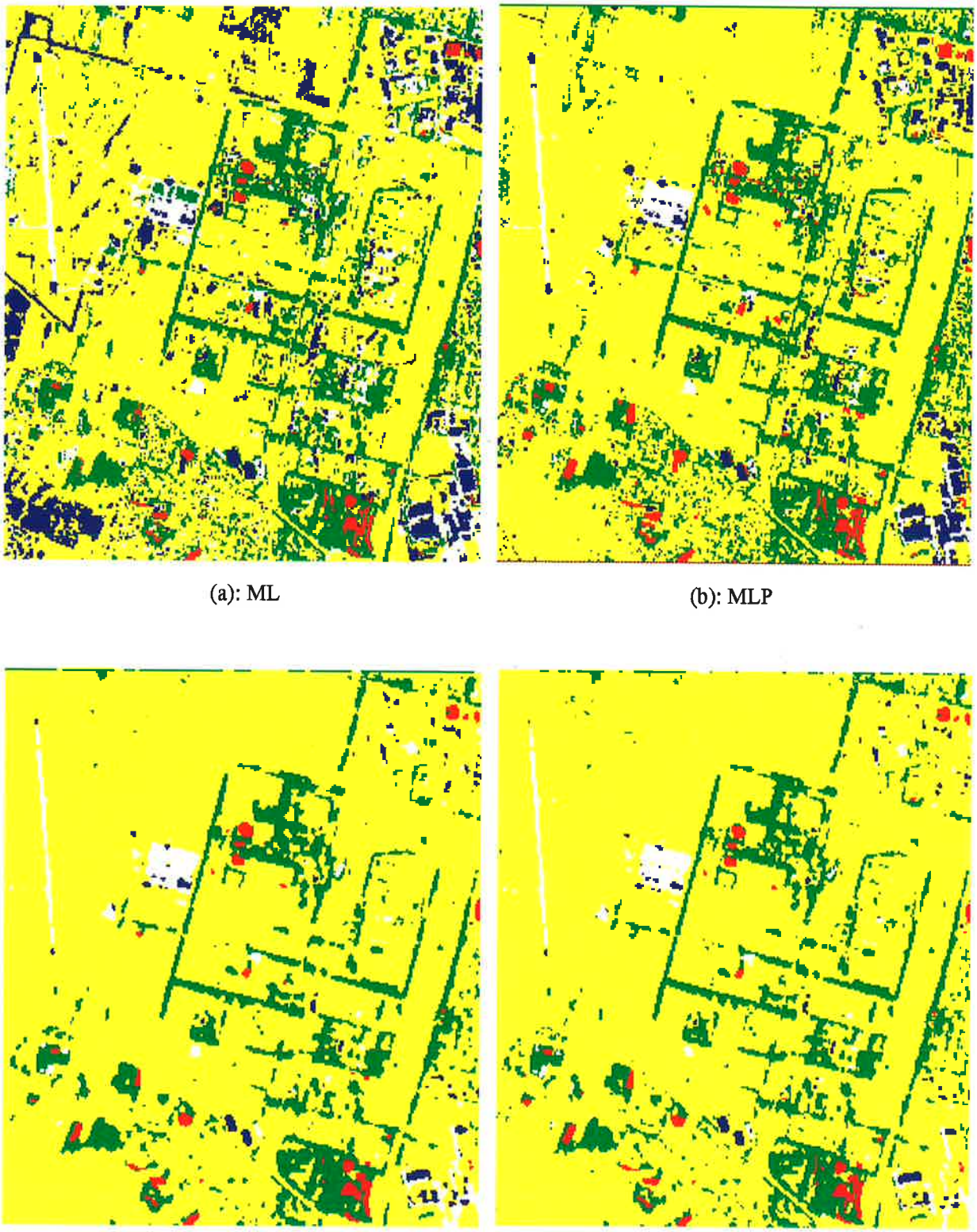
Figure 4.9: Classification of an MSS image using (a) spectral-only ML, (b) spectral-only MLP, (c) MLP with 3x3 neighbourhood, (d) MLP with clique structured neighbourhood. Blue="water", white="cloud", red="urban", green="trees/scrub", yellow="grassland"



(a): ML (b): MLP

(c): WMLP (d): CMLP

Figure 4.10: Classification of a TM image using (a) spectral-only ML, (b) spectral-only MLP, (c) MLP with 3x3 neighbourhood, (d) MLP with clique structured neighbourhood. Blue="water",white="sand/quarry", red="urban", green="trees/scrub", yellow="grassland".



(a): ML (b): MLP

(c): WMLP (d): CMLP

Figure 4.11: Classification of a SPOT image using (a) spectral-only ML, (b) spectral-only MLP, (c) MLP with 3x3 neighbourhood, (d) MLP with clique structured neighbourhood. Blue="roof", white="pavement", red="greens/ovals", green="trees", yellow="grassland".

4.3.4 Discussion

From the summary of experimental results it can be seen that the relative merits of the 3 different MLP structures depend on the data, but it is generally true to say that the extra neighbourhood data improves classification accuracy. For MSS data, the gains in using a WMLP over an MLP are obvious and there is a further increase in accuracy using clique structured inputs. For the TM data, inclusion of spatial data in the input to the MLP clearly improves classification accuracy, but the WMLP has better accuracy than the CMLP. For the SPOT data tested, spatial data makes a small improvement with the performance of the WMLP being equal to that of the CMLP.

The use of data from a moving window containing A pixels takes A times as many operations to simulate the first layer of an MLP. With the first active layer generally having the most nodes, the increase in time to simulate is significant. In ANN hardware there is no time penalty for inclusion of extra inputs but there is a penalty in the complexity of interconnection required between inputs and the first active layer's nodes. By using the CMLP structure there is a gain in speed of simulation which translates to a significant decrease in complexity in a hardware implementation of the MLP. Hence where CMLP and WMLP have equivalent accuracy, the CMLP structure should be favoured. In other cases a small reduction in accuracy from using WMLP to CMLP could be justified by the reduction in complexity of the required circuit or in a simulator, a considerable increase in speed.

The internal operation of the MLPs using neighbourhood data was examined using the "activation map" tool discussed in Chapter 3. The activation for the first active layer of CMLP reveals that nodes with similar shape cliques have similar activation maps. For example, the activation maps for the four 2x2 neighbourhood subsets have similar maps. This suggests that multiple instances of the same clique shape in different positions relative to the centre pixel may be redundant, but further exploration of this was left for future work. The existence of distinctly different activation maps for different shaped cliques is evidence that the network is able to collect different information for different shaped cliques.

Does the moving window measure texture-like information? The simple preliminary experiment demonstrated that the WMLP has the potential to use microstructure when no spectral information is present. Repetition of the experiment using the CMLP gave results similar to the WMLP. However, these experiments describe situations where spectral information has been reduced to binary information; for simplicity of image generation, pixels were black or white. The MLP is sensitive to class conditional probability density functions and may be able to use extra spectral information provided by the adjacent pixels as an extra, highly redundant supplement to spectral information. To test the spectral-texture discrimination power of the CMLP and WMLP a synthetic image was constructed in which one half contained real (textured) data and the other contained constant pixels set to the value of the mean vector of the real data. For the real data, a 40x40 MSS subimage of "urban" area was selected. The results of this experiment were negative. Neither the MLP nor the WMLP showed ability to discriminate between the two halves of the image. We conclude from this experiment that either the WMLP achieves its practical increase in accuracy using the spectral information from

surrounding pixels, possibly by filtering or deconvolution, or that this particular data set is unable to show how spatial information is being used. The possibility of the WMLP doing deconvolution is consistent with the sensors having an overlapping field of view. Filtering would be beneficial for data that comes from a true multivariate Gaussian, since it could reduce spectral variance.

4.4 Some comparative experiments using preprocessing

For comparison to the neighbourhood-classifying MLPs, some experiments were carried out to test the worth of using various texture feature extractors as preprocessors. The intention was that derived values would be used to augment the spectral radiance vector for each pixel and hence incorporate texture. In Chapter 2 this was done using PROC as a preprocessor and the augmented vector was classified using ML. In these experiments classification of augmented MSS data was performed by using the simple MLP. Classification of the derived features used as components of a pattern vector characterizing a pixel, without the inclusion of radiance information, was also tested. Various preprocessors were tested including PROC.

4.4.1 Multi-channel narrow band filters

It is possible to characterize texture by examining the image at different spatial scales. Witkin hypothesized that the human visual cortex extracts image features by correlating information across a range of scales [Witkin84]. This approach can potentially free texture description from the problems of scale mentioned earlier, but in practice it is necessary to select the *range* of scales for consideration.

Band-pass filters can be chosen to effectively extract the features of an image at various scales. In the experimentation here, band pass filters were set up to filter a single band of the image to be classified, producing new images that contained only features at the scale of interest. To filter the single band image, it was transformed into a Fourier domain frequency-magnitude image. For nine frequencies, 0...8 cycles per image, the frequency domain image was multiplied by a circularly symmetric Butterworth filter of first order to select a particular frequency component and then the image was inverse-transformed. The 9 filtered components were combined into a 9 band image.

Experiments were carried out to classify MSS data augmented with 9 bandpass filtered components based on band 6. Classification of the augmented vector and a vector consisting of only the filtered components was carried out using ML and MLP techniques. In all cases the measured accuracies of MLP classification were significantly higher than ML classifications. 0% error was reported using texture alone for some trials, confirming that the test and training areas do have sufficient texture to be useful. However, on examination of the resultant images this result was found to be misleading (figure 4.12a). The bandpass filtering leads to an image in which the components are based on large neighbourhoods producing smoothing of values. By choosing the test areas that have homogeneous land covers, results are advantaged by smoothing. Further, by choosing the test and training areas adjacent to each other, it is only necessary for a classifier to learn to classify correctly in the vicinity of the training set. The

complete classified image generated from band-pass features has class boundaries that are unacceptably blurred with very low effective resolution and extensive misclassification outside the test and training sets.

The use of bandpass filtering to generate spatial features in the way described above is, of course a non-local process. A just comparison with the 3x3 neighbourhood is not possible because the bandpass features for each pixel use the whole image as neighbourhood. An implementation that uses a small support set is required and this requirement is not met by DFT-based process[1]. The dilemma of making position-accurate estimates of local spatial frequency is also of interest in time/frequency measurements in communications and radar systems. Wavelets show considerable promise for spectral estimation in that area [Hlawatsch92], and have an analogous application in the identification of local spatial frequency in images.

4.4.2 Co-occurrence matrix based features

Haralick's co-occurrence matrix features [Haralick73] provide a computationally tortuous but conceptually simple approach to measuring texture features where textures are relatively random in nature. These features (given in table 1.1) are computationally intensive because at every pixel location in the image they require the generation of a co-occurrence matrix and then calculation of derived features. There are some obvious short cuts for calculation when using small windows, but even with short cuts, a set of texture features derived from a 3x3 window using one displacement value is very slow to calculate (needing an overnight run on a Sun Sparc2, for a 512x640 image). Unser's less computationally demanding approximations to the features [Unser86] were also tested. They gave a considerable speed-up but compromised accuracy for speed and produced consistently worse results that Haralick's features.

Experiments were conducted using the nine Haralick features of table 1.1 generated from band 6 of Landsat MSS (band 3 in the newer labelling terminology). The displacement value used for texture feature generation was (1,1) which gives the longest radial length from the centre pixel within a 3x3 window. Texture features for a displacement of (1,0) were also produced, but predictably yielded poorer results when classified. Multi-band images created by combining the nine features into one image for the first test, and for the second test by combining three features, angular second moment, entropy and homogeneity with four MSS bands. The MLP structure used was N_0-25-6-5 with N_0-determined by the input data set. Accuracies presented for the MLPs represent the best results over 10 trials starting with different weight initializations.

With the combined texture-MSS image classification accuracy on the test set was measured as 96% for the MLP, but the ML classifier classified all pixels into a single class, caused by the need to invert a covariance matrix with near zero eigen-values. Using the texture-only image

[1] Also, the implementation is inconvenient because it needs either square data of size 2^a, for arbitrary a, or else the Fourier transform must be implemented by a method other than the conventional DFT.

accuracy measured on the test set was 98% using the MLP indicating some contradictory information was derived from the MSS data.

Two examples of complete image classification using Haralick's texture features are provided in figure 4.12. Overall, the classifications are not significantly different to the spatial MLP classifications, and it is doubtful that the considerable extra processing to generate the texture features is justified.

The implementation described here calculated features from a single band of the MSS image. With the overhead of extra calculation, it would be straightforward to calculate features for all bands, although spectral interdependence would not be captured.

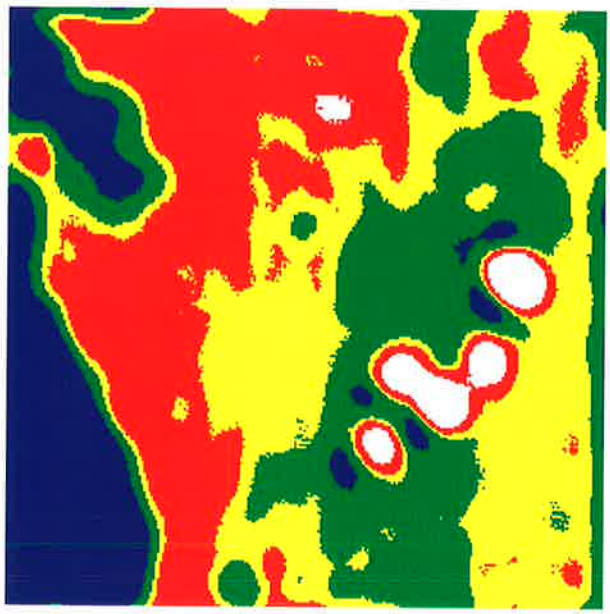
4.4.3 Spectral covariance based features

The PROC algorithm is described in Chapter 2 as a method of capturing the spectral interrelationships in a neighbourhood. In this section the class distances are used as features to augment the input vector of radiances, in a similar way to the hybrid classifier system described in Chapter 2. PROC is basically a non-linear preprocessor and produces a features with distributions that are hard to model. The five features that were generated using a 7x7 window on the primary MSS image in Chapter 2 were used again here. When applying the ML-hybrid classifier it was found necessary to scale the distance values to make them useful. In this application, the MLP is used as an adaptive classifier and it was expected that adaptation would include learning the appropriate scaling.

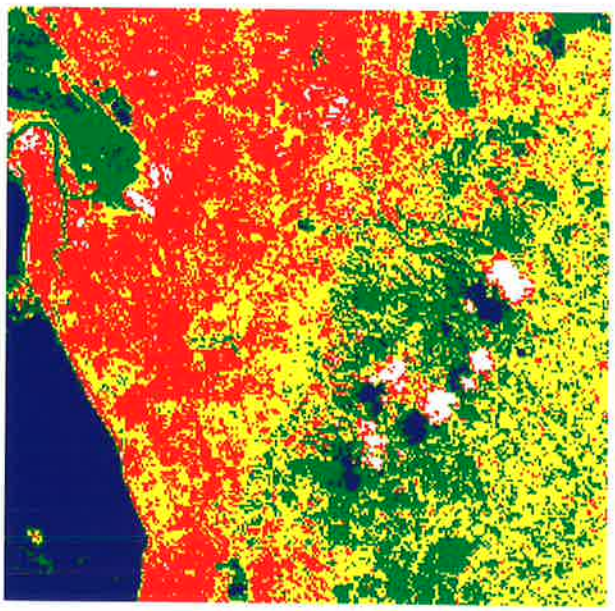
Both the ML and MLP classifiers were tested using the unscaled distance features alone and in combination with the MSS data. An MLP of structure 5-25-6-5 achieved 2% error for the distance features alone, compared to 30% error for ML[1]. This is a convincing example of an MLP being able to learn some feature-dependent information (the scaling factor) without manual intervention. For the distance-MSS combined image, an MLP achieved 3% error and the ML method gave 6%. The reduction in performance of the MLP with more information implies some of the extra information was contradictory.

An example of overall image classification is provided in figure 4.12d. Problems of performance assessment were also encountered using this pre-processor. The example shown the figure is from the classification of augmented MSS data. This has a higher effective resolution and more consistent classifications than the classification based on texture information alone, but compares poorly when assessing the classification of the test data set.

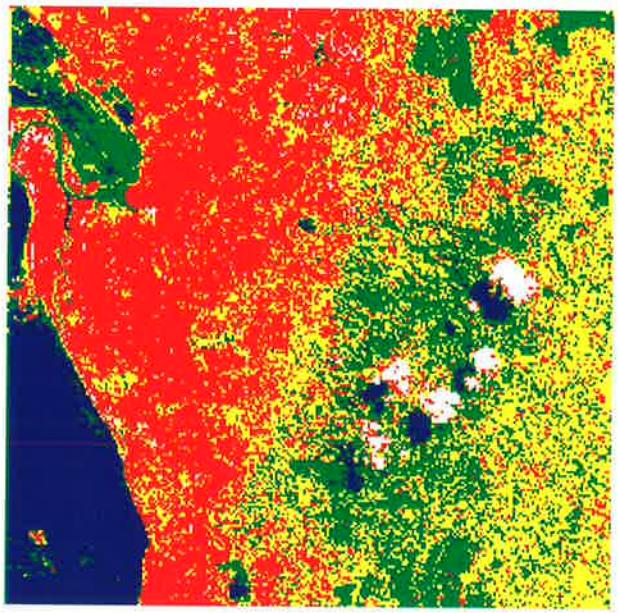
[1]Earlier results in chapter 2 were better than his by virtue of the use of scaled PROC features.



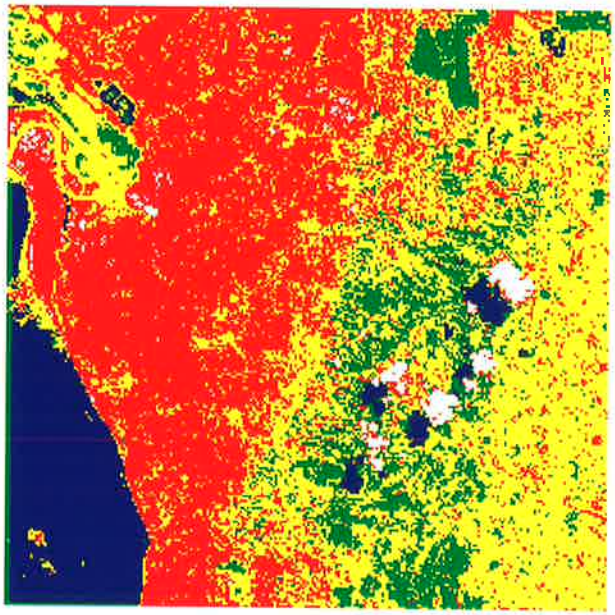
(a) Bandpass filter pre-processor



(b) 9 Haralick features from a single band



(c) 3 Haralick features + MSS



(c) 5 PROC features + MSS

Figure 4.12: Examples of MLP classification of MSS images using augmented input vectors. Blue="water", white="cloud", red="urban", green="trees", yellow="grassland".

4.4.4 Discussion of the use of preprocessors

The use of preprocessors to give texture information means that the process of classification is no longer model free. The derivation of the texture features is, in most cases, based on a model of the nature of texture. For Haralick's texture features and for the bandpass features, the underlying model is Julesz's early model of the perception of textures by humans. For PROC processing the model is as discussed in Chapter 2.

Preprocessors are clearly advantageous for increasing the accuracy of MLP (and ML) based classification systems. However, in general, preprocessing is computationally expensive. Additionally, most preprocessors need to have large neighbourhoods to be effective. Large neighbourhoods give a reduction in effective resolution in the output image.

The apparent accuracy of some of these methods when measured using training set/test set methodology was overturned by a qualitative examination of the classification of a whole image. Further examination suggests that what is required to assess accuracy is a set of reference pixels, randomly distributed throughout the image, with known classifications. The division of the reference set into test and training sets should result in two sets with equal spatial diversity. Of course, to train and test classifiers using neighbourhoods, it is not sufficient to collect data from random isolated individual pixels. To be useful, every reference pixel must be accompanied by a set of measurements that cover each pixel in its neighbourhood. Such a reference set would be difficult to collect but is not impractical.

The need for a minimum number of training samples per class, based on statistical considerations, has been well recognized from almost the beginning of work on supervised multi-spectral image classification. It has also been recognized that training sets which have samples from multiple areas for a single class are desirable, but the reason usually given is to ensure adequate diversity in the sample being used to estimate class parameters. Diversity of training data functions to broaden the variance of a class's distribution function and hence admit pixels with radiance vectors at greater distance from the centroid in spectral space (see [Duda73]).

When using a classifier such as the MLP which approximates the required mapping from input to output using non-linear functions, the concept of what constitutes "nearness" in input (spectral) space becomes complicated by the fact that different parts of input space can be treated in different ways. When data is preprocessed, because the class conditional data model is often unknown or difficult to measure, the concept of pixel similarity is further obscured. It is possible that a classifier may learn the training data exactly as a mapping from particular points in spectral space to particular classes, but with no generalizing power. In this context it is important to use an independent test set to measure accuracy, whereas in a model based classifier it is reasonable, although optimistic, to assess accuracy by examining how well the training data is characterized by the *a priori* model with the estimated parameters. This was the reasoning behind the design of experiments with separate test and training sets. Using a single reference set divided into two partitions to give the training and test sets was done for convenience.

When using spatial data, we implicitly expect that the structure of space is such that Euclidean geometry governs concepts such as "nearness". When using preprocessors or input structurings that cause inclusion of spatial data, it is possible for an adaptive classifier such as the MLP to learn to give a mapping from a spatial location to a class that is not necessarily generalizable to nearby pixels. We want the MLP to learn a class conditional texture function that is valid throughout the image, not simply at certain points. When devising the MLP structures with neighbourhood inputs, the use of a (3x3) moving window was adopted because it virtually guaranteed against learning "position" in the image. But with a preprocessor such as a bandpass filter that uses the whole image as neighbourhood, this is not the case.

The conclusion from this is that the methodology for measuring accuracy is valid for the classification systems using small moving windows but the methodology is flawed when assessing classifiers using preprocessors with large or whole-image neighbourhoods. Hence it is valid to compare the results of the 3x3-window Haralick-feature-preprocessor with the neighbourhood-input MLPs, but it is not valid to make comparisons with the bandpass preprocessor or preprocessing using PROC.

4.6 Summary and conclusions

This chapter has described multi-layer perceptrons that utilize spatial information to improve the accuracy of multi-spectral classification.

Two multi-layer perceptrons with modified architectures were presented in which spatially related information from a neighbourhood was used to classify a particular pixel. The MLPs were fed neighbourhood data for the pixel to be classified, and were expected to develop their own internal representation of "texture" from the examples. The inclusion of spatial data improved accuracy at the expense of complexity and hence, for the simulated networks, speed. An interesting result was that restricting the interconnections between inputs and nodes of the first active layer to impose "clique" structure on the spatial information does not significantly affect accuracy while producing a considerable saving in complexity of interconnection, a major consideration in implementation of real neural networks. It is conjectured that the reduction in information to individual nodes is counteracted by building in to the network a concept of spatial adjacency.

MLPs were also tested as classifiers in combination with various pre-processors. Preprocessors improve classification accuracy, but are computationally expensive and reduce output image resolution when used with large windows. Accuracies achieved in classifying test sets appeared good but assessment of the overall image classification showed that the preprocessors produced little overall improvement over the MLPs with spatial inputs. What was highlighted was a failure in the method of assessing accuracy when used on classification systems which are essentially model free and have the capacity to learn position in an image.

This problem inhibited fair comparison of the two approaches to including spatial information in a spectral classification: i.e using the MLP to learn spatial relationships without guidance *vs* capturing spatial relationships by preprocessing the image into features which are

classified by the MLP. A hand-waving assessment indicates that the spatial MLPs have less overhead than preprocessors, which is demonstrably the case where classification is a more frequent task than training, but appears data dependent in the case that relearning is frequently necessary.

This chapter has focussed on including spatial information in spectral classifications by extra processing at the input to a classifier. Each process effectively imposes a model of the input image on the data. In the next chapter we explore the possibility of imposing a model of the expected output image.

Table 4.2: Comparison of ML, MLP, WMLP, CMLP for MSS data

Classifier	Trial	Class1 ac.(%)	Class2 ac.(%)	Class3 ac.(%)	Class4 ac.(%)	Class5 ac.(%)	Max class error (%)	Average error (%)
ML		100.0	96.5	95.0	93.5	84.5	15.5	6.1
MLP	1	100.0	100.0	90.5	97.0	70.0	30.0	8.5
MLP	2	100.0	100.0	90.5	97.0	71.0	29.0	8.3
MLP	3	100.0	100.0	96.0	93.5	89.5	10.5	4.2
MLP	4	100.0	100.0	89.5	96.0	70.5	29.5	8.8
MLP	5	100.0	100.0	97.5	93.0	88.0	12.0	4.3
MLP	6	100.0	100.0	97.0	93.5	88.0	12.0	4.3
MLP	7	100.0	100.0	91.0	97.0	68.0	32.0	8.8
MLP	8	100.0	100.0	96.5	93.5	88.5	11.5	4.3
MLP	9	100.0	100.0	90.0	97.0	70.0	30.0	8.6
MLP	10	100.0	100.0	89.5	97.0	73.5	26.5	8.0
WMLP	1	100.0	100.0	100.0	98.5	99.5	1.5	0.4
WMLP	2	100.0	100.0	100.0	98.5	99.5	1.5	0.4
WMLP	3	100.0	100.0	100.0	98.5	99.5	1.5	0.4
WMLP	4	100.0	100.0	100.0	92.5	100.0	7.5	1.5
WMLP	5	100.0	100.0	100.0	98.5	99.5	1.5	0.4
WMLP	6	100.0	100.0	100.0	98.5	100.0	1.5	0.3
WMLP	7	100.0	100.0	100.0	98.5	99.5	1.5	0.4
WMLP	8	100.0	100.0	100.0	98.5	99.5	1.5	0.4
WMLP	9	100.0	100.0	100.0	98.5	99.5	1.5	0.4
WMLP	10	100.0	100.0	100.0	98.5	100.0	1.5	0.3
CMLP	1	100.0	100.0	100.0	99.0	99.0	1.0	0.4
CMLP	2	100.0	100.0	100.0	100.0	78.5	21.5	4.3
CMLP	3	100.0	100.0	100.0	100.0	77.0	23.0	4.6
CMLP	4	100.0	100.0	100.0	99.0	99.5	1.0	0.3
CMLP	5	100.0	100.0	100.0	99.5	97.5	2.5	0.6
CMLP	6	100.0	100.0	99.5	100.0	94.5	5.5	1.2
CMLP	7	100.0	100.0	100.0	98.5	99.5	1.5	0.4
CMLP	8	100.0	100.0	100.0	99.0	98.5	1.5	0.5
CMLP	9	100.0	100.0	100.0	99.5	99.5	0.5	0.2
CMLP	10	100.0	100.0	100.0	99.0	98.5	1.5	0.5

Table 4.3: Comparison of ML, MLP, WMLP, CMLP for TM data

Classifier	Trial	Class1 ac. (%)	Class2 ac.(%)	Class3 ac.(%)	Class4 ac.(%)	Class5 ac.(%)	Max class error (%)	Average error (%)
ML		100.0	91.8	94.9	90.3	87.6	12.4	7.1
MLP	1	100.0	95.9	98.9	93.2	88.9	11.1	4.6
MLP	2	100.0	95.6	99.3	95.4	86.2	13.8	4.7
MLP	3	100.0	92.5	99.5	84.0	78.4	21.6	9.1
MLP	4	100.0	91.8	99.6	86.7	83.9	16.1	7.6
MLP	5	100.0	93.5	99.9	83.2	81.9	18.1	8.3
MLP	6	100.0	94.2	99.3	86.0	84.5	15.5	7.2
MLP	7	100.0	93.2	99.2	93.6	87.9	12.1	5.2
MLP	8	100.0	95.9	99.8	91.5	81.4	18.6	6.3
MLP	9	100.0	92.5	99.2	80.6	80.7	19.4	9.4
MLP	10	100.0	94.9	99.1	93.9	88.1	11.9	4.8
WMLP	1	100.0	94.9	99.6	99.6	95.2	5.1	2.2
WMLP	2	100.0	94.9	100.0	94.0	75.7	24.3	7.1
WMLP	3	100.0	95.9	99.1	98.7	96.4	4.1	2.0
WMLP	4	100.0	93.8	99.9	98.2	92.2	7.8	3.2
WMLP	5	100.0	92.5	100.0	97.6	76.2	23.8	6.8
WMLP	6	100.0	95.2	100.0	90.6	56.8	43.2	11.5
WMLP	7	100.0	92.8	100.0	89.4	55.8	44.2	12.4
WMLP	8	100.0	90.8	100.0	94.5	70.9	29.1	8.8
WMLP	9	100.0	95.9	100.0	89.5	77.9	22.1	7.4
WMLP	10	100.0	95.6	100.0	95.1	80.8	19.2	5.7
CMLP	1	100.0	85.6	99.3	96.8	96.2	14.4	4.4
CMLP	2	100.0	88.0	99.5	96.2	95.6	12.0	4.1
CMLP	3	100.0	89.0	99.6	93.8	94.1	11.0	4.7
CMLP	4	100.0	90.8	99.5	99.4	91.4	9.3	3.8
CMLP	5	100.0	92.5	99.7	94.5	86.8	13.2	5.3
CMLP	6	100.0	91.8	99.8	95.3	87.7	12.4	5.1
CMLP	7	100.0	87.7	99.6	92.4	89.7	12.3	6.1
CMLP	8	100.0	91.8	99.8	93.6	82.8	17.2	6.4
CMLP	9	100.0	90.1	99.4	95.1	96.9	9.9	3.7
CMLP	10	100.0	88.7	99.4	94.4	96.6	11.3	4.2

Table 4.4: Comparison of ML, MLP, WMLP, CMLP for SPOT-XS data

Classifier	Trial	Class1 ac. (%)	Class2 ac. (%)	Class3 ac. (%)	Class4 ac. (%)	Class5 ac. (%)	Max class error (%)	Average error (%)
ML		65.2	50.7	98.1	94.4	83.4	49.3	21.7
MLP	1	65.2	77.7	100.0	96.2	87.1	34.9	14.8
MLP	2	63.6	76.9	100.0	97.2	86.3	36.4	15.2
MLP	3	65.2	76.9	100.0	95.3	87.9	34.9	15.0
MLP	4	66.7	77.3	100.0	95.8	87.7	33.3	14.5
MLP	5	63.6	76.9	100.0	97.2	86.1	36.4	15.2
MLP	6	63.6	76.9	100.0	97.2	86.6	36.4	15.1
MLP	7	65.2	76.9	100.0	95.8	89.6	34.9	14.5
MLP	8	63.6	76.9	100.0	96.7	86.3	36.4	15.3
MLP	9	66.7	76.9	100.0	95.8	87.3	33.3	14.7
MLP	10	65.2	78.6	100.0	95.8	87.4	34.9	14.6
WMLP	1	59.1	76.0	100.0	94.8	100.0	40.9	14.0
WMLP	2	62.1	74.7	100.0	94.4	100.0	37.9	13.8
WMLP	3	60.6	74.7	100.0	94.4	100.0	39.4	14.1
WMLP	4	62.1	76.0	100.0	92.5	100.0	37.9	13.9
WMLP	5	60.6	74.7	100.0	92.0	100.0	39.4	14.5
WMLP	6	62.1	76.0	100.0	94.4	100.0	37.9	13.5
WMLP	7	59.1	76.0	100.0	94.8	100.0	40.9	14.0
WMLP	8	60.6	76.0	100.0	94.4	100.0	39.4	13.8
WMLP	9	60.6	74.7	100.0	92.5	100.0	39.4	14.5
WMLP	10	60.6	76.0	100.0	94.8	100.0	39.4	13.7
CMLP	1	59.1	75.6	98.1	94.8	98.3	40.9	14.8
CMLP	2	59.1	76.9	100.0	95.8	98.5	40.9	14.0
CMLP	3	60.6	75.1	100.0	91.1	99.7	39.4	14.7
CMLP	4	60.6	73.4	100.0	93.4	98.5	39.4	14.8
CMLP	5	63.6	75.1	100.0	93.4	99.0	36.4	13.8
CMLP	6	60.6	74.7	100.0	90.6	99.5	39.4	14.9
CMLP	7	60.6	74.7	100.0	92.0	99.9	39.4	14.6
CMLP	8	57.6	74.7	100.0	90.1	99.9	42.4	15.5
CMLP	9	60.6	74.2	100.0	92.0	99.7	39.4	14.7
CMLP	10	62.1	76.4	100.0	94.8	99.0	37.9	13.5

Chapter 5

MLPs using Spatial Context

Summary: *This chapter presents an MLP that uses spatial context to improve classification of multi-spectral images. Previous chapters have shown that an MLP can achieve classification accuracy comparable to the classical equal-priors maximum likelihood (ML) scheme on single pixel inputs. Besag's "iterated conditional modes" technique can be used to iteratively refine the conventional ML classification; class priors for ML classification can be constructed by using the previous estimate of classified output image to define a distribution on the class variation in a neighbourhood of a given pixel. This chapter describes how a similar approach can be implemented on an MLP. The training strategy, which has an important bearing on the results, is also discussed.*

5.1. Overview

This chapter presents an artificial neural net classifier that improves spectral classification by the inclusion of information available from *spatial context*. We define *spatial context* to be the spatial relationship of a pixel's class to the classes of other pixels near it. A human image interpreter can detect potential mis-classifications by reference to context; for example, an isolated "tree" label surrounded by "water" labels will need stronger evidence to avoid being rejected than an isolated "tree" surrounded by "grassland". Use of context to improve classification accuracy is inherently an activity that takes place either after classification or else as part of an iterative classification scheme, since to assess how a pixel's label relates to that of its neighbours requires at least a rough estimate of the neighbours' classes.

Spatial context has been used to improve conventional methods of classification in the field of remote sensing. Richards *et al.* presented a sophisticated post processing scheme based on pixel relaxation [Richards81]. Swain, Vaderman and Tilton used compound decision theory to develop a method for incorporating prior information about the probability of allowable neighbourhood configurations into the classification process [Swain81]. Their method required an estimate of a "context function" which is a set of probability density functions for each possible neighbourhood configuration. Approximations of the context function made from a non-context classification provided promising results, but accurate estimation proved to be a difficult problem.

Besag [Besag86], and later Kiiveri and Campbell [Kiiveri91], pursued methods of iterative classification refinement under the names of iterated conditional modes (ICM) and cyclic ascent algorithm (CAA). In their approaches, both the input image, comprising (multivariate) measurements, and the output image, comprising labels, are characterized as Markov random fields. This follows the approach of Geman and Geman [Geman84] with monochromatic images. Besag discussed an intuitively appealing variant of ICM where properties of the output image are expressed by probabilities of certain classes adjoining each other in a particular orientation. To implement this method it is necessary to specify primitives to measure "oriented" adjacency. Primitives can only be devised by detailed inspection of the desired output image.

This chapter proposes a scheme modelled after Besag's ICM that can be implemented using an ANN. Previous chapters have shown that an MLP can achieve accuracy comparable to the classical equal-priors ML scheme on single-pixel multi-spectral data. Using an MLP as the classifier, a scheme is presented in which a classification is based not only on the multi-spectral measurements for a pixel but also on the results of an earlier estimate of classified output. The architecture described here, while influenced by the recursive networks of Elman [Elman90], is unique in the field of MLPs for image processing.

Experiments to test the context classifier are difficult to perform without specialized (expensive) ground truth. The context classifier must learn to accept, rather than filter, sharp transitions from one class to another, but to learn this kind of behaviour, examples of transitions from one class to another must be present in the training set. Ground truth data collected for use with conventional classifiers is usually not suitable since examples are generally collected from the middle of known single-class areas to avoid erroneous labelling.

Some limited experiments using synthetic images are provided here to show that the MLP with context can be used to improve the accuracy of a non-context scheme. However, the amount of improvement is influenced strongly by the rather limited model used to generate the synthetic data. Further work will be required to validate the method on real data.

5.2. Iterative schemes for inclusion of context

5.2.1 A review of a scheme using conventional ML classification

This section briefly reviews ICM in the notation of Chapter 1.

ICM allows the incorporation of context information into conventional ML scheme by the use of the class prior, which is, of course, disregarded in "equal priors" schemes. The prior term is designed to incorporate information about the class labels of the pixels in the neighbourhood of the pixel to be classified.

Recall from Chapter 1 that a "classified image" is an image consisting of class labels $Z = \{z_{kl} \mid (k,l) \in L, z_{kl} \in \Omega\}$ where $z_{kl} = \omega_r \Rightarrow x_{kl}$ comes from class ω_r. (L is the lattice which defines location within the image and $\Omega = \{\omega_r \mid r = 1,\ldots R\}$ is the set of all possible

classes.) Within a Bayesian framework, a pixel (k,l) with measured value x_{kl} is assigned a label z_{kl} that satisfies (1.9):

$$\max_{z_{kl} \in \Omega} [P(x_{kl}|z_{kl})\, P(z_{kl})]$$

Taking logs for convenience, this becomes

$$\max_{z_{kl} \in \Omega} [\log P(z_{kl}) + \log P(x_{kl}|z_{kl})] \tag{5.1}$$

If we model the classes as multivariate normal distributions with means m_r and covariance matrices C_r, then under the assumption that the probabilities are dependent on class but not position, we can write:

$$z_{kl} = \max_{\omega_r \in \Omega} [\log P(\omega_r) - \log|C_r| - (x - m_r)^t C_r^{-1} (x - m_r)] \tag{5.2}$$

where some class-independent constants have been neglected. Deleting the first term, which is the prior probability of any pixel belonging to a particular class, we are left with the familiar equal priors ML discriminant function (1.11).

The derivation of (5.2) relies on two assumptions stated in Chapter 1: assumption A1 is that the measurements from pixels are independent, and assumption A2 is that pixels' (true) labels are independent. If we accept that the spatial context in a image contains useful information, it is necessary to reject the assumption that the (true) pixel labels are independent. The most general case is that each pixel label t_{kl} depends on all the other pixels' labels in the image, written $T \backslash t_{kl}$, but this leads to an intractable classification problem.

If we instead postulate that T, the image of true (but unknown) class labels, is a second order Markov random field, then it is possible to incorporate this prior information about the classified image into the classification process by manipulation of the prior term in (5.2) [Besag86]. A suitable selection of prior probability is

$$P(z_{kl}) = \frac{1}{a} \exp\left\{ \sum_{r=1}^{R} \alpha_r n_r(k,l) + \sum_{r=1}^{R} \sum_{s=1}^{r} \beta_{rs} n_{rs}(k,l) \right\} \tag{5.3}$$

where $n_r(k,l)$ is the number of pixels of class ω_r in η_{kl}, the neighbourhood of (k,l) excluding (k,l) itself, $n_{rs}(k,l)$ is the number of adjacent pixel pairs of classes ω_r and ω_s in η_{kl} a is a normalization constant, and α and β are parameters. A further decomposition of the second term is possible in which there is a unique parameter for each pixel pair at each possible location in the neighbourhood. A useful practical implementation of (5.3) has a separate parameter for each possible orientation of pixel pairs [Kiiveri91]. Geman and Geman used a different parameter for each clique set of pixels [Geman84].

Clearly, the extraction of a simple discriminant function is not possible, since to measure n_r and n_{rs} we need to know the true classification in advance. ICM uses $\hat{\mathbf{Z}}$, the current estimate of $\boldsymbol{T}$, to estimate the neighbourhood counts needed to construct the next estimate, $\boldsymbol{Z}$. Using estimates of the neighbourhood counts, and substituting for the prior term in (5.2), a suitable rule for the labelling of pixel (k,l) is:

$$z_{kl} = \max_{\omega_r \in \Omega} \left[\sum_{r=1}^{R} \alpha_r \hat{n}_r(k,l) + \sum_{r=1}^{R} \sum_{s=1}^{r} \beta_{rs} \hat{n}_{rs}(k,l) - \log|C_r| - (x - m_r)^t C_r^{-1} (x - m_r) \right] \tag{5.4}$$

Besag discusses schemes in which individual pixels or groups of pixels have their current estimated labels updated immediately and warns that convergence is not guaranteed for "synchronous" update.

Implementation of ICM as a modification to a conventional ML classifier is not difficult but the parameters introduced in 5.3 are data dependant. Besag provides an example in which he sets $\alpha = 0$ and $\beta_{r,s} = \beta = 1.5 \ \forall \ r,s$. Kiiveri and Campbell use pseudo-likelihood estimation with tuning[1] and Geman and Geman use heuristics based on energy considerations. More general estimation of these parameters is difficult and outside the scope of this thesis.

5.2.2 An MLP-based system

In previous chapters, it has been established that an MLP can be used to perform the same classification function as an ML classifier. Hence, to classify a multi-spectral image, an MLP must effectively implement a set of R discriminant functions of the form (1.11), although it uses a completely different parameterization of the problem.

The architecture presented here is designed to incorporate the extra information needed by the prior term of (5.4) into an MLP classifier. This extra information is the labelling of the neighbourhood of the pixel to be classified. It is information that we would expect a human operator to use, albeit, unwittingly. The MLP is expected to learn from this extra information appropriate contexts for each of the classes. Of course, there is no real parallel between this form of artificial intelligence and human intelligence. A human is able to use much more prior information than just information about the structure of images. A human's recognition of objects can impose useful constraints on the image interpretation from other domains of learning which cannot be replicated by the limited training of the MLP.

The proposed system (figure 5.1) of classification uses an extended MLP that has additional inputs to allow the labels of the neighbours of the pixel to be classified to be taken into account. The classification process is iterative and stops when the output image is stable for one iteration. There is no requirement to model the output image beyond the Markov random field assumption that spatial dependency of the true class labels is limited to the 2nd order neighbourhood.

[1] i.e. the value was modified by human intervention.

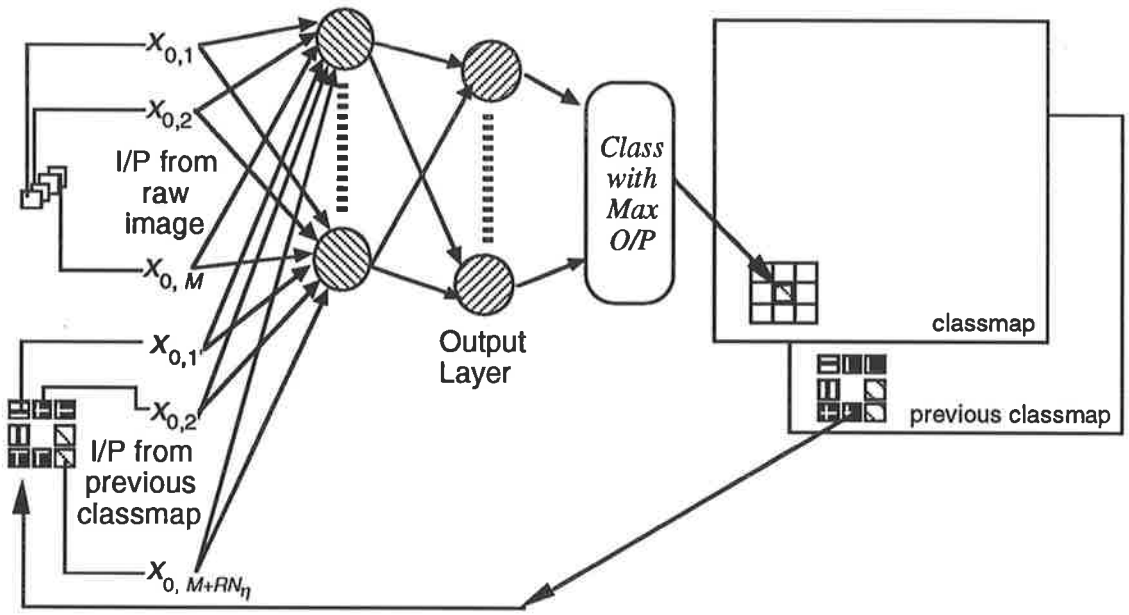


Figure 5.1: An MLP for context incorporation.

In previous chapters, the term *order* has been used to describe the neighbourhood of the pixel and, in particular, to define how many extra pixels from the input image are to be taken into account as contributors to the vector of *measurements* to be classified by the MLP. In the scheme proposed in this chapter estimated *labels* in a neighbourhood of the pixel being classified provide additional contribution to the MLP's input vector. These labels must come via a feedback path from a previous output image. It is feasible, and in experiments below, usual, that the neighbourhood used to collect input radiance data can be different to the neighbourhood used to collect labels. This makes it necessary to distinguish an *input* order and an *output* order.

5.3. Design issues

5.2.1 Representing the labelled neighbourhood

The information that we want to feed back comprises a neighbourhood of labels. The experiments described in the next section use an output order of 2, which limits the number of labels taken into account to $N_\eta = 8$ labels (The previously estimated label of the pixel to be classified is not used). The position relative to the pixel to be classified is important and must be preserved. This suggests an architecture with N_η extra inputs to the MLP, each getting a class number.

But, it is misleading to represent the labels by simple numerical values; to do so would imply that classes whose labels have small numeric differences are similar. The only un-biased representation possible is $N_\eta.R$ inputs (for R classes). A further reason for using this arrangement is that it allows us to examine weights given to the feedback connections in order to assess their contribution to the classification power of the network.

The use of a large number of inputs to capture context information is costly in terms of speed of simulation, translating to complexity of implementation in hardware. Given enough examples and large enough network, an MLP could theoretically learn to adapt to the bias caused by simply feeding back class labels. As a compromise, it may be possible to find a more elegant function of the labels to reduce the number of inputs required, without implying a nonexistent relationship. In particular cases, biases induced by non-ideal functions of this kind may be acceptable to reduce the size of the MLP.

5.3.2 Training procedure

The procedure used to train this system is governed strongly by the complexity (and hence time for simulation) that is acceptable. An obvious way to train such a system is to perform a classification for each pixel, iterating until stability before deriving the error signal for adjusting the weights. However, while this approach may be feasible using an MLP realized in hardware, the amount of time required to simulate such a system means that it is not feasible to study it in software. Further, there is no evidence that the intermediate states during training would lead to a stable converging system at every stage during training.

An alternative training strategy was developed in which the extra context inputs are derived from the desired output image, rather than a previous iteration. Weight adjustments are made after each presentation of a training pixel, according to the same schedule as the MLPs of previous chapters. A single training example comprises the multivariate measurement for the pixel to be classified, the desired label, and the desired labels of the neighbouring pixels. It is an open question whether this kind of training leads to convergence during classification without a good starting estimate of the output. Modifications may be necessary.

5.3.3 Selection of training areas

To train a classifier that uses spatial context, the training set must allow the classifier to learn a variety of multi-class label arrangements such as edges, adjacencies, encirclements. Hence, the type of training data required for extended MLP architecture is an image containing a mixture of all the required classes, not in distinctly separate areas, but with each pixel correctly labelled. Such training sets of any reasonable size are rare.

5.4 Experiments

All tests presented here are on carefully structured synthetic data. The synthetic data were based on a real 4-band Landsat MSS image that contained five identifiable classes. Five synthetic classes were modelled as multivariate normal distributions with the class parameters (mean and covariance) measured from training sets in the real image. To generate a synthetic image, the real image's ML classified output image was used as a template. For each pixel at location (i,j) a 4-component vector equivalent to an MSS radiance vector was constructed by generating a random vector from the multivariate normal distribution described by the class parameters of the class label at location (i,j) in the template. In this way, the multivariate normal model of the class density functions is guaranteed and the true classification of every

pixel is known. Further, the spatial relationships between class labels in the real data are preserved in the synthetic data.

While ideally class membership should be approximately equal, in practice such a test set is difficult to find. For the test data used here, the class memberships were: ω_1=2316; ω_2=75; ω_3=4142; ω_4=2530; ω_5=41.

Two alternative MLP structures were tested in separate experiments. Each experiment comprised a (1) classification using spectral data alone, (2) a classification using context from the true labels and (3) a classification using the estimated classification from the non-context MLP as the context information.

The MLP structures consisted of 4 inputs plus context inputs, 50 hidden units, and 5 output units. Following earlier work, the learning rate was chosen to be 0.3, momentum was chosen to be 0.0, and the stopping criterion was 50,000 presentations of patterns from each class. In the first experiment, 40 context inputs were used, each being a boolean valued input for every possible label at each (output) neighbourhood position. This structure provides the MLP with unbiased information about the labelled neighbourhood. The second experiment was carried out to gauge the effect of the bias caused by feeding back the actual (arbitrarily assigned) class numbers. Eight context inputs were used, each fed with a (scaled) class number corresponding to the labelling of the estimated neighbourhood.

Classifications were carried out on an MLP simulator written in C and running under the Khoros image processing environment [Rasure90]. Times taken for the training and classification process ranged from 5 to 50 minutes using a Sun Sparc2. No ICM comparison is included here because of the difficulty of making appropriate estimates for the parameters α and β_{rs}.

5.5 Results

Results from these limited experiments are encouraging. Summary results are provided in table 5.1 and detailed classification accuracies are given in contingency tables 5.2-5.6 (at the end of this chapter). Table 5.2 gives results of classifications with no context information. Tables 5.3 and 5.5 contain the results of classifications given a context of the true labels.

Not surprisingly, the classification accuracy of a simple pixel-classifying MLP is surpassed by the accuracy of an MLP with knowledge of the correct classifications of neighbouring pixels. More usefully, tables 5.4 and 5.6 give the accuracies when the context information is the estimated classification from MLPs corresponding to table 5.2.

The result of experimenting with the obvious but biased method of feeding back raw class labels confirms that this method is unsuitable. The 12-input MLP was only able to improve classification accuracy marginally with the same amount of training as the 44-input MLP.

Table 5.1: Summary of results of context experiments.

Classifier	Context	Extra inputs	Max error in any class	Average error
MLP 4-50-5	none	none	14.0%	6.6%
MLP 44-50-5	true	40 units	4.4%	2.8%
MLP 44-50-5	estimate	40 units	5.8%	3.0%
MLP 12-50-5	true	8 units	11.3%	4.8%
MLP 12-50-5	estimate	8 units	11.0%	5.4%

The images used in these tests are given in figure 5.2. Figure 5.2-a shows the template used for synthetic image generation, figure 5.2-b shows the classification using spectral data alone (i.e. no context) and figure 5.2-c shows the result of including a non-context estimate of the labelled image as the context for another stage of MLP classification. Errors in the no-context image are due to the MLP incorrectly classifying pixels with values taken from the tails of the class-conditional density function. The context classifier cleans up these errors, but not by a simple smoothing action that would blur the classification, even where the membership of a class is well defined.

The conclusions that can be drawn from the tests reported here are limited because the data used was artificial and relatively easy to classify without context. Further, a potential problem with an MLP-based context classifier is that it cannot escape from local minima during the iterative process. Wright addressed this problem by using simulated annealing [Wright89], but at a grossly increased cost in terms of computational requirement. Local minima were not found to be a problem in the extension of conventional classifiers to handle context reviewed above [Kiiveri91]. It will only be possible to state confidently that an MLP in this role will not have serious problems after considerable further experimentation.

5.6 Summary and conclusions

Given that an MLP is able to preform multi-spectral image classification, it is useful to investigate any methods for increasing accuracy that may have been applied to conventional

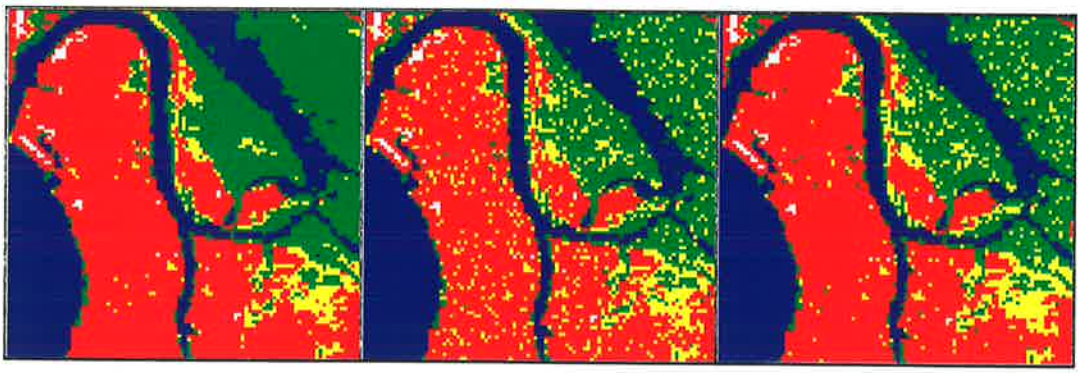


Figure 5.2: Class label images: (a) Template; (b) Classified with no-context; (c) Classified with context.

methods of classification. This chapter has considered accuracy increase by inclusion of spatial context information.

Two MLP architectures that can use spatial context have been presented. In some simple experiments the use of context was shown to improve classification accuracy on synthetic images. Further testing with real data is desirable but will require extensively ground-truthed images.

Table 5.2: Contingency table for classification without context. MLP structure: 4-50-5.

True Class	Assigned Class				
	1	2	3	4	5
1	100.00	0.00	0.00	0.00	0.00
2	0.00	98.46	0.00	0.00	1.54
3	0.00	0.08	85.86	0.10	13.96
4	0.00	0.00	0.00	87.55	12.45
5	0.00	0.00	0.00	5.05	94.95
Average classification accuracy 93.36%					

Table 5.3: Contingency table for classification with context comprising true labels. MLP: 44-50-5.

True Class	Assigned Class				
	1	2	3	4	5
1	100.00	0.00	0.00	0.00	0.00
2	0.00	100.00	0.00	0.00	0.00
3	0.00	0.43	96.37	0.05	03.14
4	0.00	0.00	0.00	95.60	04.40
5	0.00	0.00	1.94	03.88	94.17
Average classification accuracy 97.23%					

Table 5.4: Contingency table for classification with context comprising estimated labels. MLP: 44-50-5.

True Class	Assigned Class				
	1	2	3	4	5
1	99.90	0.00	0.00	0.10	0.00
2	0.00	100.00	0.00	0.00	0.00
3	0.00	0.08	98.36	0.05	1.51
4	0.00	0.00	0.04	96.21	3.74
5	0.00	0.00	5.83	03.69	90.49
Average classification accuracy 96.99%					

Table 5.5: Contingency table for classification with context comprising true labels. MLP: 12-50-5.

True Class	Assigned Class				
	1	2	3	4	5
1	100.00	0.00	0.00	0.00	0.00
2	0.00	98.67	0.00	1.33	0.00
3	0.00	0.05	96.98	0.24	2.73
4	0.00	0.00	0.00	88.70	11.30
5	0.00	0.00	2.59	5.73	91.68
Average classification accuracy 95.21%					

Table 5.6: Contingency table for classification with context comprising estimated labels. MLP: 12-50-5.

True Class	Assigned Class				
	1	2	3	4	5
1	100.00	0.00	0.00	0.00	0.00
2	0.00	98.67	0.00	1.33	0.00
3	0.00	0.07	94.18	0.31	5.43
4	0.00	0.00	0.00	89.01	10.99
5	0.00	0.00	1.66	7.21	91.13
Average classification accuracy 94.60%					

Chapter 6

Conclusions and Summary

6.1 Summary of Results

The goal of the work reported in this thesis was to develop a methodology to classify multi-spectral images using spatial information, but without separating the image into monochromatic subimages. The reason to avoid splitting the image into separate single band images is to allow measurement of the spatial arrangement, at the micro-structure level, of (multi-spectrally) coloured entities. This micro-structure of coloured entities has been called multi-spectral texture.

In developing a methodology for this kind of multi-spectral image classification, the goals were to use all the spectral bands and if possible to use the information contained in the inter-spectral band structure that is disregarded when image bands are considered separately.

6.1.1 PROC for image spectral structure exploitation

One way to exploit the class-conditional inter-band information in multispectral images is to make use of the spectral covariance matrix of pattern (radiance) vectors in a neighbourhood. Of course, the ML classifier uses the covariance matrix to scale the distance to a class's centroid in pattern space. Using the covariance matrix itself as a characteristic of a class is rare. The covariance matrix is generated from a demeaned pattern vector, and hence if the majority of power to distinguish between classes comes from knowledge of the mean of the pattern vector, discrimination between classes based on covariance alone would be poor. Chapter 2 shows that by using the neighbourhood spectral covariance matrix as a pixel's attribute, the similarity measuring algorithm, "pattern recognition by observation correlations" can be used to produce a classification of an MSS image that is at least comparable to a classification obtained using the benchmark ML scheme. However, PROC has problems at edges and where patches of a class are smaller than the moving window size. In these circumstances pixel covariance matrices are unlike any reference class covariance matrix and the assigned class is the result of comparing large uncertain values of a "distance".

To try to use the information provided by PROC in a more stable form, a hybrid pattern vector which could be classified using ML was generated. For each pixel a vector was constructed from the multispectral radiance data for that pixel plus transformed PROC "distances" (D_r) from the pixel's neighbourhood covariance to each class covariance. Classification using the hybrid pattern vector was shown to be superior to classification using a pattern vector of either D_r's alone or multi-spectral data alone. In Chapter 2, considerable effort was expended to devise a transformation, and also some *ad hoc* class biases, to make the D_r into features suitable for ML classification. The transformation process was needed to make the density function of transformed D_r approximate a normal distribution. The possibility of avoiding this transformation lead to an interest in distribution-free classifiers, and in particular artificial neural networks. In Chapter 4 unbiased D_r were used in a successful MLP-based classifier. The utility of PROC is somewhat reduced by the amount of computation required to extract covariance similarity measures from image and its practical application could only be justified when other methods failed.

The PROC algorithm was originally of interest because it showed promise for the transfer of spectral signatures from one image. PROC was at least able to make a reasonable classification of a second image compared to a poor classification by the ML classifier, but the resulting accuracies are not sufficient for a routine use. Classification of areas of the ground affected by cloud shadow was not shown to be viable due to the loss of data resolution for low values caused by the sensor system's A/D conversion quantization step size; i.e. dynamic range considerations.

6.1.2 An MLP for multi-spectral image classification

Chapter 3 established the usefulness of MLPs for the classification of multi-spectral data without inclusion of spatial data, as a preparation for the following chapters. With suitable training, a neural network based on a three layer perceptron is theoretically capable of classifying patterns from sets with arbitrarily distributed density functions in N-dimensional space. Thus, potentially, an MLP can classify with greater accuracy than a model based classifier when the underlying structure of the model is not a good approximation to the structure of the data. This is undoubtedly the case in many applications where it is standard practice to use equal-priors maximum likelihood estimation based on multi-variate normal distribution.

It was established by experiment that by using back-propagation of error, it is possible to train an appropriately structured MLP to learn class charateristics in MSS data. A large number of MLP structures were tested for "trainability". Lack of training convergence due to local minima in the energy function, a common problem of steepest descent approximation methods (which include BP), did not appear to be a problem in this particular data domain. It was also observed that with a carefully selected structure it is possible for an MLP to exceed the accuracy of ML by a considerable margin on MSS data. More usefully, with a conservatively oversized MLP it is possible to routinely exceed the classification accuracy of ML, given a good selection

of weights during the training phase. The structures chosen for subsequent tests represented conservative rather than optimal approaches to MLP structure.

For a particular structure, the quality of training, as measured by the performance of the trained MLP on an unseen test set, was found to be highly dependent on the randomly selected initial weights. Hence, to be reasonably confident that an MLP will classify accurately, weights need to be selected after repeated trials in which weights are initialized to different values. The issue of how many trials are required to select a good set of weights needs further research, but ten trials were observed to be acceptable for this kind of data A training strategy in which a number of different sets of weights is developed and tested is not extravagant when the amount of data to be classified is large and a single classification is fast, which would be the case with appropriate hardware.

A problem, characteristic of learning systems with emergent properties, is the inscrutability of the trained MLP. Some insight into the internal representation in MLPs can be provided by visualization techniques. If we model the classifying action of an MLP after Lippmann's explanation (figure 3.6), then it is desirable to see each node in the first active layer partitioning input pattern hyperspace into half spaces (that are combined by nodes in later layers). This is easy to portray with a two dimensional input pattern space but with 4, 7 or more dimensions it is quite difficult . For 4-band MSS data a system was devised in which pattern space (ie. input data) was projected into a two dimensional space whose axes corresponded to the principal components of all the training data[1]. The input-space-partitioning hyperplanes defined by the MLP's first layer weights were also projected into the new space. Hyperplanes were observed to cross the principal component axis of the training data almost at right angles which is a desirable orientation for good class discrimination.

Activation maps were introduced to probe the operation of the MLPs for an appropriate set multi-spectral data. These maps show a relationship between active nodes' responses and the required input-to-output transfer function. The maps from the output layer gave an indication of the certainty of classification. Examination of the hidden layers' activation maps suggested that an MLP acts as a non-linear adaptive filter. It is also possible to detect potential redundancies in the networks from the activation maps of hidden layers. However, extracting evidence for particular internal representation within a trained network remains a difficult task.

6.1.3 An extended MLP to include texture

Ideally, an artificial neural network should be able to discover the concept of texture without aid if it is trained with all data from a neighbourhood. In Chapter 4, an MLP using information from a 3x3 neighbourhood, was shown to give an improvement in classification accuracy over the simple MLP approach, but at the cost of much greater network complexity, and hence for simulated networks, speed. A more efficient scheme was proposed that used spatially structured but otherwise un-preprocessed data as input to the neural network. This scheme

---

[1] This transformation can provide an illuminating view of the clustering of multiviarate data, although the benefits of such a transformation are highly data dependent.

restricted the interconnection between neighbourhood inputs and the first active layer to try to provide a hint to the network about what constituted spatial adjacency. The MLPs that used a pixel's data plus data from particular shapes from within the pixel's neighbourhood ("cliques") achieved a significantly reduction in the complexity of the network with a marginal change in classification accuracy; for some data there was a marginal improvement in accuracy although this appeared to be specific to the MSS reference sets.

MLPs were also tested as classifiers in combination with various pre-processors which were used to generate texture features for each pixel. Preprocessors improve classification accuracy, but are computationally expensive, reduce output image resolution when used with large windows, and require an *a priori* model of texture. A problem of classifying preprocessed data is that the act of preprocessing can change the data distribution to be non-normal distribution. Hence an MLP classifier has potentially higher accuracy that the model based ML classifier. Notably, the MLP proved to be a significantly better classifier of the PROC-augmented hybrid vector from Chapter 2.

For the three preprocessors tested, accuracies achieved in classifying test sets were excellent but assessment of the overall image classification was disappointing; clearly the positioning of the test sets near the training sets was critical. It was noted that model-free classification systems which have spatial information as part of their input have, at least in part, the capacity to learn position. This lowered the value of the test sets used in the experiments and inhibited fair comparison of the preprocessor based MLPs and the spatial input MLPs. A qualitative assessment indicated that the spatial MLPs have less overhead than preprocessors when classification is a more frequent task than training, but the choice of method is not clear when relearning is frequently necessary.

6.1.4 An extended MLP to include context

Given that an MLP is able to preform multi-spectral image classification, it is useful to investigate any methods for increasing accuracy that may have been applied to conventional methods of classification. Chapter 5 presented an MLP that used spatial context to improve classification of multi-spectral images. An architecture was proposed in which classification accuracy was iteratively refined by using the previous estimate of classified output image to define a distribution on the class variation in a neighbourhood of a given pixel. A training strategy, which has an important bearing on the results, was proposed and tested on limited synthetic data. Further testing with real data is desirable but will require extensively ground-truthed images.

6.2 Contributions

Chapter 2 developed a method of covariance comparison due to Bogner [Bogner81a] into an algorithm that could be applied to the classification of pixel neighbourhoods. The chapter provided some new results about the distribution of the proposed metric and an original implementation of the technique for use in images.

Chapter 3 provided a thorough investigation of use of multi-layer perceptrons for the classification of spectral data in a directly analogous manner to ML. The experimental work lead to the widely suspected but elsewhere unvoiced conclusion that it is necessary to repeat the training of a network from different starting points in order to be even reasonably sure of a sensible final trained network. Some original insights were provided into the internal representation of a successfully trained network using eigen transformations of data and hyperplanes and by the introduction of activation maps.

Chapter 4 explored methods of integrating spatial information into a neural network classifier. An efficient network presented involves the unique structuring of input connections using clique concepts from Markov random field theory. In some ways this network is a "frozen" version of Geman and Geman's image restoration procedure [Geman84], but used in a different problem domain.

Chapter 5 proposed an original neural network based classifier which was designed to learn context. Some simple empirical results were given.

6.3 Future work.

The work described in this thesis was planned to be an exploration of preprocessors and classifiers to distil the essence of multi-spectral texture. In retrospect, the tools used, especially the MLP, were not as well defined or understood as a superficial understanding made them appear. The PROC covariance similarity measure was an interesting basis of neighbourhood spectral covariance, but there are other measures which could also justify investigation. The MLP was chosen after examination of its claimed properties, but details of implementation in the literature were sparse. Because of this, it is intended that the material in Chapter 3 should serve as a guide for anyone wishing to build a successful multi-spectral image classifying MLP. (The computer programs used for simulations and preprocessing can be made available by the author).

It is a truism that full confidence in the reliable application of MLPs awaits further theoretical results. Current theoretical results on MLPs for such quantities as nodes per layer, number of layers and learning rate are only available as bounds, which are generally not useful in practical situations. More analysis along the lines pursued by Levin *et al.* [Levin90] is eagerly awaited.

However, despite this lack of theoretical proof, the MLP certainly can function as a classifier and can have an accuracy better than conventional methods. In situations where a large amount of data is to be classified, and where appropriate hardware is available, it may not be unreasonable to train an ML and a number of MLP classifiers, compare accuracy on a validating test set, and choose the best classifier. Given an MLP in hardware, speed of classification will always be an advantage with a large number of bands.

Further work is also justified on the use of alternative classifiers with derived texture-feature vectors. The work exploring the use of MLP classifiers with preprocessed data was encouraging, but used only three potential texture feature generators. Other feature generators may be useful and the use of other non-parametric classifiers may also be advantageous.

Assessment of performance of any classifier using spatial context essentially needs a set of standard images, each with defined training and test sets. Ideally standard images would range from 4-band images to 256-band images. Some standard (multi-spectral) colour texture images would also be useful to test colour texture algorithms. It appears that there is no widely accepted standard for colour, with Brodatz's texture [Brodatzt66] being the *de facto* standard for testing monochromatic images.

The process of attempting to use all multispectral bands is essentially a process of fusing data from the various bands. Since this process is quite general an MLP similar to those used in this work will be suitable for fusing that data to provide enhanced information in situations where data is sampled on a regular 2D grid. Fusing of information for target tracking is subject of much research, but data for target tracking is not usually sampled on a regular grid. While no experimental evidence is provided here, an MLP should be able to learn spatial relationships on any grid, given all the normal requirements of enough training data, enough attempts to learn from random starting points and an appropriate structure.

6.4 Summary

This thesis has explored some aspects of multi-spectral texture.

Matching of covariance matrices proved to be a promising new way of classifying multi-dimensional image data, but required some *ad hoc* choices and modifications to produce an algorithm. Best results were obtained when used in a hybrid system in conjunction with a MLP classifier.

When classifying a multi-spectral image with multi-variate normal distributed classes, an MLP can closely approximate the performance of the optimal maximum likelihood scheme. With more pathologically distributed class probability density functions, an MLP's classification accuracy can exceed that of the ML scheme.

Given the good performance of an MLP, it is reasonable to extend its application to include spatial as well as spectral information in ways analogous to the extensions to standard ML. However, with the unique learning potential of a MLP, in particular its ability to develop new internal representations of the problem space, it has been possible to explore more flexible extensions to a simple spectral classifier. Some of these implementations have been shown to be useful.

In all cases there is a trade-off between resources required for each classifier and the achievable increase in classification accuracy. The resources are the amount of time required to train a classifier and the complexity of the (simulated) computer architecture. An underlying assumption of this work has been that ANN hardware will be developed for general use and that it will be widely available because it will not need to be specialized for a particular task. As new hardware becomes available, different aspects of the trade-offs will become relevant and it will be appropriate to apply the techniques examined in this thesis in diverse applications.

Bibliography

Aach88. T. Aach, U. Franke, and R. Mester, "From Texture Energy Measures to Quadrature Filter Pairs - A System Theoretical View of Texture Feature Extraction," in *Signal Processing IV: Theories and Applications*, J.L. Lacoume, A. Chehikian, N. Martin, and J. Malbos, Eds. Elsevier Science Publishers B.V., 1988, pp. 827–830.

Ade83. F. Ade, "Characterization of Textures by 'Eigenfilters'," *Signal Processing*, vol. 5, pp. 451–457, 1983.

Barber91. D.G. Barber and E.F. LeDrew, "SAR Sea Ice Discrimination Using Texture Statistics: A Multivariate Approach," *Photogrammetric Engineering and Remote Sensing*, vol. 57, no. 4, pp. 385–395, Apr. 1991.

Benediktsson89. J.A. Benediktsson, P.H. Swain, and O.K. Ersoy, "Neural Network Approaches versus Statistical Methods in Classification of Multi-Source Remote Sensing Data," in *Proc. International Conference on Geoscience and Remote Sensing*, Vancouver, July 1989, pp. 489–492.

Benediktsson90. J.A. Benediktsson and P.H. Swain and O.K. Ersoy, "Neural Network Approaches Versus Statistical Methods in Classification of Multi-Source Remote Sensing Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 540–552, July 1990.

Benediktsson91. J.A. Benediktsson, O.K. Esroy, and P.H. Swain, "A Consensual Neural Network," in *Proceedings of International Conference on Geoscience and Remote Sensing*, Espoo, Finland, June 1991, pp. 2219–2222.

Besag86. J. Besag, "On the Statistical Analysis of Dirty Pictures," *Journal Royal Statistical Society, Series B*, vol. 48, no. 3, pp. 259–302, 1986.

Blanz90. W.E. Blanz and S.L. Gish, "A Connectionist Classifier Architecture Applied to Image Segmentation," Tech. Rept., IBM Almaden Research Centre, San Jose, CA. 95120., 1990.

Bogner81. R.E. Bogner, "On Talker Verification Via Orthogonal Parameters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, (reprint), Feb. 1981.

Bogner81a. R.E. Bogner, "Pattern Recognition via Observation Correlations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 2, pp. 128–133, Mar. 1981.

Bovik90. A.C. Bovik, M. Clark, and W.S. Geisler, "Multichannel Texture Analysis Using Localized Spatial Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 55–73, Jan. 1990.

Bovik91. A.C. Bovik, "Analysis of Multichannel Narrow-Band Filters for Image Texture Segmentation," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2025–2043, Sep. 1991.

Bridle89. J.S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," in *Neuro-computing: algorithms, architectures and applications*, F. Fougelman-Soulie and J. Hérault, Eds. Berlin: Springer-Verlag, 1989.

Brodatz66. P. Brodatz, *Textures – A Photographic Album for Artists and Designers.* New York: Dover, 1966.

Buf90. J.M.H. du Buf, M. Kardan, and M. Spann, "Texture Feature Performance for Image Segmentation," *Pattern Recognition*, vol. 23, no. 3, pp. 291–309, 1990.

Burrascano91. P. Burrascano, "A Norm Selection Criterion for the Generalized Delta Rule," *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 125–130, Jan. 1991.

Burt83. P.J. Burt, "Fast Algorithms for Estimating Local Image Properties," *Computer Graphics and Image Processing*, vol. 21, pp. 368–382, 1983.

Carpenter87. G.A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, vol. 26, no. 23, pp. 4919–4930, Dec. 1987.

Chellappa85. R. Chellappa and S. Chatterjee, "Classification of Textures Using Gaussian Markov Random Fields," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 33, no. 4, pp. 959–963, Aug. 1985.

Chui91. W.C. Chui and E. Hines, "A Rule-Based Dynamic Back-Propagation (DBP) Network," in *Proc. IEE International Conference on Neural Networks*, Bournemouth, UK., Nov. 1991, pp. 170–173.

Cohen91. F.S. Cohen, Z. Fan, and M.A. Patel, "Classification of Rotated and Scaled Textured Images Using Gaussian Markov Random Field Models," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 13, no. 2, pp. 192–202, Feb. 1991.

Conners80. R.W. Conners and C.A. Harlow, "A Theoretical Comparison of Texture Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 3, pp. 204–222, May 1980.

Crippen87. R.E. Crippen, "The Regression Intersection Method of Adjusting Image Data for Band Ratioing," *International Journal of Remote Sensing*, vol. 8, no. 2, pp. 137–155, 1987.

Cross83. G.R. Cross and A.K. Jain, "Markov Random Field Texture Models," *IEEE Transactions on Pattern Recognition and Machine Analysis*, vol. 5, no. 1, pp. 25–39, Jan. 1983.

Cushine87. J.L. Cushine, "The Interactive Effect of Spatial Resolution and Degree of Internal Variability within Land-Cover Types on Classification Accuracies," *International Journal of Remote Sensing*, vol. 8, no. 1, pp. 15–29, 1987.

Cybenko89. G.D. Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303-314, Apr. 1989.

Daugman89. J.G. Daugman, "Relaxation Neural Network for Complete Discrete 2-D Gabor Transforms," in *Proc. 6th Scandinavian Conference on Image Analysis*, Oulu, Finland, June 1989, pp. 33–46.

Daugman90. J.G. Daugman, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 7, pp. 1169–1179, July 1988.

Derin86. H. Derin and W.S. Cole, "Segmentation of Textured Images Using Gibbs Random Fields," *Computer Vision, Graphics and Image Processing*, vol. 35, pp. 72–98, 1986.

Derin87. H. Derin and H. Elliot, "Modelling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 39–55, Jan. 1987.

Derin89. H. Derin and P.A. Kelly, "Discrete-Index Markov-type Random Processes," *Proceedings of the IEEE*, vol. 77, no. 10, pp. 1485–1510, Oct. 1989.

deSouza82. P. de Souza, "Texture Recognition via Autoregression," *Pattern Recognition*, vol. 15, no. 6, pp. 453–464, 1982.

Devijver82. P.A. Devijver and J.V. Kittler, *Pattern Recognition: A Statistical Approach.* Englewood Cliffs: Prentice Hall, 1982.

Dozier88. J. Dozier, "HIRIS - NASA'S High-Resolution Imaging Spectrometer for the Earth Observing System," *IEEE Geoscience and Remote Sensing Newsletter*, pp. 17–21, Nov. 1988.

Duda73. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley and Sons, 1973.

Eklundh86. J.O. Eklundh, A. Lansnet, and R. Wessblad, "Classification of Multispectral Images using Associative Nets," in *Proc. 8th International Conference on Pattern Recognition*, Paris, Oct. 1986, pp. 1240–1243.

Elman90. J.L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

Engel83. J.L. Engel. and O. Weinstein, "The Thematic Mapper — An Overview," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 21, no. 3, pp. 258-265, July 83.

Fahlman88. S.E. Fahlman, "An Empirical Study of Learning Speed in Back Propagation Networks," Tech. Rept. No. CMU-CS-88-162, Carnegie Mellon University, June 1988.

Fahlman90. S.E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky, Ed, Morgan Kaufmann, 1990, (preprint).

Franklin89. S.E. Franklin and D.R. Peddle, "Spectral Texture for Improved Class Discrimination in Complex Terrain," *INt. Journal of Remote Sensing*, vol. 10, no. 8, pp. 1437–1443, Aug. 1989.

Freund71. J.E. Freund, *Mathematical Statistics.* New Jersey: Prentice-Hall, 2nd Ed., 1971.

Fu76. K.S. Fu, "Pattern Recognition in Remote Sensing of the Earth's Resources," *IEEE Transactions on Geoscience Electronics*, vol. 14, no. 1, pp. 10–18, Jan. 1976.

Fu86. K.S. Fu, "Syntactic Pattern Recognition," in *Handbook of Pattern Recognition and Image Processing*, T.Y. Young and K.S. Fu, Eds. San Diego, CA: Academic Press Inc., 1986, pp. 85–117.

Fukushima86. K. Fukushima, "A Neural Network Model for Selective Attention in Visual Pattern Recognition," *Biological Cybernetics*, vol. 55, pp. 5–15, 1986.

Gagalowicz81. A. Gagalowicz, "A New Method for Texture Fields Synthesis: Some Applications to the Study of Human Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 5, pp. 520–533, Sep. 1981.

Gagalowicz86. A. Gagalowicz, S.D. Ma, and C. Tournier-Lasserve, "Efficient Models for Color Textures," in *Proc. 8th International Conference on Pattern Recognition*, Paris, Oct. 1986, pp. 412–414.

Galatsanos89. N.P. Galatsanos and R.T. Chin, "Digital Restoration of Multichannel Images," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 3, pp. 415–421, Mar. 1989.

Gallant88. S.I. Gallant, "Connectionist Expert Systems," *Communications of the ACM*, vol. 31, no. 2, pp. 152–169, 1988.

Geman84. S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, Nov. 1984.

Gish89. S.L. Gish and W.E. Blanz, "Comparing a Connectionist Trainable Classifier with Classical Statistical Decision Analysis Methods," Tech. Rept., IBM Almaden Research Centre, San Jose, CA,95120., June 1989.

Gish90. S.L. Gish and W.E. Blanz, "Comparing the Performance of Connectionist and Statistical Classifiers on an Image Segmentation Problem," Tech. Rept., IBM Almaden Research Centre, San Jose, CA,95120., Jan. 1990.

Gorman88. R.P. Gorman and T.J. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets," *Neural Networks*, vol. 1, pp. 75–89, 1988.

Gotlieb90. C.C. Gotlieb and H.E. Kreyszig, "Texture Descriptors Based on Co-occurrence Matrices," *Computer Vision Graphics and Image Processing*, vol. 51, pp. 70–86, 1990.

Granrath81. D.J. Granrath, "The Role of Human Visual Models in Image Processing," *Proceeding of the IEEE*, vol. 69, no. 5, May 1981, pp.552–561.

Grossberg87a. S. Grossberg and G.A. Carpenter, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 54–115, Jan. 1987.

Hampshire90. J.B. Hampshire II and B.A. Pearlmutter, "Equivalence Proofs for Multi-Layer Perceptron Classifiers and the Bayesian Discriminant Function," in *Proc. 1990 Connectionist Models Summer School*, San Mateo, CA., D. Touretzky, E. Elma, T. Sejnowski, and G.E. Hinton., Eds, Morgan Kaufman, 1990.

Haralick73. R.M. Haralick, K. Shanmugan, and I. Dinstein, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610–621, Nov. 1973.

Haralick74. R.M. Haralick and K. Shanmugan, "Combined Spectral and Spatial Processing of ERTS Imagery Data," *Remote Sensing of the Environment*, vol. 3, pp. 3–13, 1974.

Haralick79. R.M. Haralick, "Statistical and Structural Approaches to Texture," *Proceedings of IEEE*, vol. 67, pp. 786–804, May 1979.

Haralick83. R.M. Haralick, "Decision Making in Context," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 5, pp. 417–428, July 1983.

Haralick86. R.M. Haralick and H. Joo, "A Context Classifier," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 24, no. 6, pp. 977–1007, Nov. 1986.

Hartt89. K. Hartt, M.J. Carlotto, and M.W.B. Brennan, "A Method for Multi-Dimensional Image Segmentation," in *Proc. International Conference on Geoscience and Remote Sensing*, Vancouver, July 1989, pp. 509–512.

He90. D.C. He and L. Wang, "Texture Unit, Texture Spectrum, and Texture Analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 509–512, 1990.

Hecht-Nielsen90. R. Hecht-Nielsen, *Neurocomputing*, Reading, MA: Addison-Wesley, 1990.

Hepner90. G.F. Hepner, T. Logan, N. Ritter, and N. Bryant, "Artificial Neural Network Classification Using a Minimal Training Set: Comparison to Conventional Supervised Classification," *Photogrammetric Engineering and Remote Sensing*, vol. 56, no. 4, pp. 469–473, Apr. 1990.

Hinton86. G.E. Hinton and T.J. Sejnowski, "Learning and Re-learning in Boltzmann Machines," in *Parallel Distributed Processing*, D.E. Rumelhart and J.J. McClelland, Eds. MIT press, 1986, Ch. 7, pp. 282–317.

Hinton87. G.E. Hinton, "Learning Translation Invariant Recognition in a Massively Parallel Networks (sic)," in *Proceedings of PARLE 87, part 1* . Springer Verlag, 1987, pp. 1–13.

Hlawatsch92. F. Hlawatsch and G.F. Bourdeaux-Bartels, "Linear and Quadratic Time-Frequency Representations," *IEEE Signal Processing Magazine*, vol. 9, no. 2, pp. 21–67, Apr. 1992.

Hopfield86. J.J. Hopfield and D.W. Tank, "Computing with Neural Circuits: A Model," *Science*, vol. 233, pp. 625–633, Aug. 1986.

Horn86 . B.H.P. Horn, *Robot Vision*, Cambridge, MA: MIT Press, 1986.

Hornik89. K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

Hornik90. K. Hornik, M. Stinchcombe, and H. White, "Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks," *Neural Networks*, vol. 3, pp. 551–560, 1990.

Hsiao90. J.Y. Hsiao and A.A. Sawchuk, "Supervised Textured Image Segmentation Using Feature Smoothing and Probabilistic Relaxation Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 12, pp. 1279–1292, Dec. 1989.

Hubel59. D.H. Hübel and T.N. Wiesel, "Receptive Fields of Single Neurons in the Cat's Striate Cortex," *Journal of the Physiology*, vol. 148, pp. 574–591, 1959.

Hunt84. B.R. Hunt and O. Kübler, "Karhunen-Loeve Multispectral Image Restoration, Part 1: Theory," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 3, pp. 592–600, June 1984.

Hurlbert88. A.C. Hurlbert and T.A. Poggio, "Learning a Color Algorithm from Examples," in *Neural Processing Systems: Proc. NIPS, 1987, Denver, Co.*, D.Z. Anderson, Ed. NY: American Institute of Physics, 1988, pp. 622–631.

Jensen79. S.K. Jensen and F.A. Waltz, "Principal Components Analysis and Canonical Analysis in Remote Sensing," in *Proc 45th Annual Meeting of the American Photogrammetric Society*, 1979, pp. 337–348.

Jensen79a. J.R. Jensen, "Spectral and Textural Features to Classify Elusive Land Cover at the Urban Fringe," *Professional Geographer*, vol. 31, no. 4, pp. 400–409, 1979.

Jordon91. R.L. Jordon, B.L. Huneycutt, and M. Werner, "The SIR-C/X-SAR Synthetic Aperture Radar System," *Proceedings of the IEEE*, vol. 79, no. 6, pp. 827–838, June 1991.

Julesz62. B. Julesz, "Visual Pattern Discrimination," *IRE Transactions on Information Theory*, vol. 8, pp. 84–92, Feb. 1962.

Julesz65. B. Julesz, "Texture and Visual Perception," *Scientific American*, vol. 212, no. 2, pp. 38–48, Feb. 1965.

Julesz73. B. Julesz, E.N. Gilbert, L.A. Shepp, and H.L. Frisch, "Inability of Humans to Discriminate Between Visual Textures that Agree in Second Order Statistics — Revisited," *Perception*, vol. 2, pp. 391–405, 1973.

Julesz86. B. Julesz, "Texton Gradients: The Texton Theory Revisited.," *Biological Cybernetics*, vol. 54, pp. 245–251, 1986.

Kamata91. S. Kamat, R. Eason, and E. Kawaguchi, "Classification of LANDSAT Image Data Using a Neural Network Approach," in *Proc. 7th Scandinavian Conference on Image Analysis*, Aalborg, Denmark, Aug. 1991, pp. 1110–1117.

Kanellopoulos91. I. Kanellopoulous, A. Varfis, G.G. Wilkinson, and J. Megier, "Classification of Remotely Sensed Satellite Images Using Multi-Layer Perceptron Networks," in *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds. Elsevier Science Publishers, 1991, pp. 1067–1070.

Kartikeyan91. B. Kartikeyan and A. Sarkar, "An Identification Approach for 2-D Autoregressive Models in Describing Textures," *CVGIP: Graphical Models and Image Processing*, vol. 53, no. 2, pp. 121–131, Mar. 1991.

Kashyap86. R.L. Kashyap and A. Khotanzad, "A Model-Based Method for Rotation Invariant Texture Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 472–481, July 1986.

Kauth76. R.J. Kauth and G.S. Thomas, "The Tassled Cap – A Graphic Description of the Spectral-Temporal Development of Agricultural Crops as Seen by Landsat," in *Proc. Symposium on Machine Processing of Remotely Sensed Data*, 1976, pp. 41–51.

Khotanzad90. A. Khotanzad and J.Y. Chen, "Unsupervised Segmentation of Textured Images by Edge Detection in Multidimensional Features," *IEEE Transactions on System Man and Cybernetics*, vol. 11, no. 4, pp. 414–421, Apr. 1989.

Kiiveri86. H.T. Kiiveri and N.A. Campbell, *Allocation of Remotely Sensed Data using Markov Models for Spectral Variables and Pixel Labels*, preprint of [Kiiveri91], referenced in [Besag86], 1986.

Kiiveri91. H.T. Kiiveri and N.A. Campbell, "Allocation of Remotely Sensed Data using Markov Models for Spectral Variables and Pixel Labels," *Australian Journal of Statistics*, (preprint), 1991.

Kittler84. J. Kittler and J. Föglein, "Contextual Classification of Multispectral Pixel Data," *Image and Vision Computing*, vol. 2, no. 1, pp. 13–29, Feb. 1984.

Klinker88. G.J. Klinker, S.A. Shafer, and T. Kanade, "Image Segmentation and Reflection Analysis Through Colour," in *Proc. SPIE Conference on Application of Artificial Intelligence IV*, 1988, pp. 229–244.

Kobatake86. H. Kobotake and J. Moroo, "Partitioning of Texture Image Using Two-Dimensional Linear Prediction Model," in *Proc. IEEE-IECEJ-ASJ International Conference on Acoustics, Speech and Signal Processing*, IEEE, Apr. 1986, pp. 1437–1440.

Kohonen87. T. Kohonen, "Adaptive, Associative, and Self-organizing Functions in Neural Computing," *Applied Optics*, vol. 26, no. 3, pp. 4910–4918, Dec. 1987.

Kolen91. J.F. Kolen and A.K. Goel, "Learning in Parallel Distributed Processing Networks: Computational Complexity and Information Content," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 347-358, Mar. 1991.

Kosko88. B. Kosko, "Bidirectional Associative Memories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 49–60, Jan. 1988.

Kowalik83. W.S. Kowalick, R.J.P. Lyon, and P. Switzer, "The Effects of Additive Radiance Terms on Ratios of Landsat Data," *Photogrammetric Engineering and Remote Sensing*, vol. 49, no. 5, pp. 659–669, May 1983.

Lang88. K.J. Lang and M.J. Witbrock, "Learning to Tell Two Spirals Apart," in *Proc. 1988 Connectionist Summer School*, Morgan Kaufmann, 1988, pp. 52–59.

Laws80. K.I. Laws, *Textured Image Segmentation*, Ph.D. dissertation, Department of Electrical Engineering, Image Processing Institute, University of Southern California, Los Angeles, California 90007, Jan. 1980.

Levin90. E. Levin, N. Tishby, and S.A. Solla, "A Statistical Approach to Learning and Generalization in Layered Neural Networks," *Proc. IEEE*, vol. 78, no. 10, pp. 1568–1574, Oct. 1990.

Li74. K.P. Li and G.W. Hughes, "Talker Differences as They Appear in Correlation Matrices of Continuous Speech Spectra," *Journal of the Acoustical Society of America*, vol. 55, no. 4, Apr. 1974.

Li89. B. Li and W.M. Moon, "Eigenvector Projection Transformation and Dimension Size Reduction in Remote Sensing Data Processing," in *Proc. International Conference on Geoscience and Remote Sensing*, Vancouver, July 1989, pp. 503–508.

Lippmann87. R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4–22, Apr. 1987.

Longstaff89. I.D. Longstaff and B.L. Reid, "Training the Perceptron with Sparse Data Sets," in *Proc. Australian Symposium on Signal Processing and Applications*, Adelaide, South Australia, Apr. 1989, pp. 240–243.

Longstaff90. I.D. Longstaff and D.L. Howard, "Image Restoration with a Locally Connected Boltzmann Machine," in *Proc. First Australian Conference on Neural Networks*, Sydney, Australia, Jan. 1990, pp. 65–66.

Lu78. S.Y. Lu and K.S. Fu, "A Syntactic Approach to Texture Analysis," *Computer Graphics and Image Processing*, vol. 7, pp. 303–330, 1978.

Lu79. S.Y. Lu and K.S. Fu, "Stochastic Tree Grammar Inference for Texture Synthesis and Discrimination," *Computer Graphics and Image Processing*, vol. 9, pp. 234–245, 1979.

Lundahl86. T. Lundahl, W.J. Ohley, S.M. Kay, and R. Siffert, "Fractional Brownian Motion: A Maximum Likelihood Estimator and its Application to Image Texture," *IEEE Transactions on Medical Imaging*, vol. 5, no. 3, pp. 142–161, Sep. 1986.

Mackay91a. D.J.C. MacKay, *A Practical Bayesian Framework for Backprop Networks*, preprint from Neuroprose archive[1], dated May 1991 .

Marceau90. D.J. Marceau, P.J. Howarth, J.M.M. Dubois, and D.J. Gratton, "Evaluation of the Grey-Level Co-Occurrence Matrix Method For Land-Cover Classification Using SPOT Imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 513–519, July 1990.

Mardia79. K.V. Mardia, J.T. Kent, and J.M. Bibby, *Multivariate Analysis*. London, UK.: Academic Press, 1979.

Marshall91. S.J. Marshall, R.F. Harrison, and R. Kennedy, "Neural Classification of Chest Pain Symptoms: A Comparative Study," in *Proc. IEE International Conference on Neural Networks*, Bournemouth, UK., Nov. 1991, pp. 200–204.

Minsky69. M.L. Minsky and S.A. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969, (1972 version with handwritten alterations).

Minsky88. M.L. Minsky and S.A. Papert, *Perceptrons: An Introduction to Computational Geometry, (expanded edition)*. Cambridge, MA.: MIT Press, 1988.

Mitchell77. O.R. Mitchell, C.R. Meyers, and W. Boyne, "A Min-Max Measure for Image Texture Analysis," *IEEE Transactions on Computers* , pp. 408–414, 1977.

Moller90. M.F. Møller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," Neural Networks, in press, 1992. (1990 preprint).

Muirhead82. R.J. Muirhead, *Aspects of Multivariate Statistical Theory*. New York: John Wiley and Sons, Inc, 1982.

Nichol90. D.G. Nichol, "Region Adjacency Analysis of Remotely-Sensed Imagery," *International Journal of Remote Sensing*, vol. 11, no. 11, pp. 2089–2101, Nov. 1990.

Nilsson65. N.J. Nilsson, *Learning Machines: Foundations of Trainable Pattern Classifying Systems*. New York: McGraw-Hill, 1965.

NTSC54. Special Issue on the NTSC Color Television Standards, *Proceedings of the IRE*, Jan. 1954.

Oja89. E. Oja, J. Parkkinen, K. Selkåinaho, and T. Kårki., "Regularity Measurement, Classification and Segmentation of Textures," *Proc. of From Pixels to Features*, pp. 207–218, 1989.

Papoulis84. A. Papoulis, *Probability, Random Variables and Stochastic Processes*. Tokyo, Japan: McGraw Hill, 2nd, 1984.

Parkkinen86. J. Parkkinen and E. Oja, "Co-occurrence Matrices and Subspace Methods in Texture Analysis," in *Proc. 8th International Conference on Pattern Recognition*, Paris, Oct. 1986, pp. 405–408.

Pech86. R.P. Pech and R.D. Graetz, "Reflectance Modeling and the Derivation of Vegetation Indices for an Australian Semi-Arid Scrubland," *Remote Sensing of the Environment*, vol. 7, pp. 389–403, 1986.

---

[1] Neuroprose is a computer archive at Ohio State University that can be publicly accessed via internet host: archive.cis.ohio-state.edu (128.146.8.52).

Peleg84. S. Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple Resolution Texture Analysis and Classification," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 6, no. 4, pp. 518–523, July 1984.

Pentland84. A.P. Pentland, "Fractal-Based Description of Natural Scenes," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 6, no. 6, pp. 661–674, Nov. 1984.

Pickard91. R.W. Picard and I.M. Elfadel, "On the Structure of Optimal Aura Matrices for the Autobinomial Markov/Gibbs Texture Model," Tech. Rept. No. 160, MIT Media Laboratory and Modelling Group, Cambridge MA, 02139, Feb. 1991.

Pickard91a. R.W. Picard and I.M. Elfadel, "Auras Part II: Application to Markov/Gibbs Texture Modelling," Tech. Rept. No. 158, MIT Media Laboratory and Modelling Group, Cambridge MA, 02139, Feb. 1991.

Poggio90. T. Poggio and F. Girosi, "Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks," *Science*, vol. 247, pp. 978–982, Feb. 1990.

Pratt71. W.K. Pratt, "Spatial Transform Coding of Color Images," *IEEE Transactions on Communication Technology*, vol. 19, no. 6, pp. 980–992, Dec. 1971.

Press86. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes*. Cambridge University Press, 1986.

Ramapriyan85. H.K. Ramapriyan, J.P. Strong, and J.C. Tilton, "The Massively Parallel Processor – Programming and Applications," in *Proc. of Pecora 10*, Colorado State University, Fort Collins, Colorado, Aug. 1985, pp. 546–555.

Rao90. A.R. Rao, *A Taxonomy for Texture Description and Identification*. New York: Springer-Verlag, Springer Series in Perception Engineering, 1990.

Rasure90. J. Rasure, Y. Williams, D.A. A., and G. Sauer, "A Visual Language and Software Development Environment for Image Processing," *International Journal of Imaging Systems and Technology*, vol. 2, pp. 183–199, Feb. 1990.

Redding91. N. Redding, A. Kowalczyk, and T. Downs, "Higher Order Separability and Minimal Unit Fan In," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and K. J, Eds, Elsevier Science Publishers, 1991, pp. 25–30.

Reed90. T.R. Reed and H. Wechsler, "Segmentation of Textured Images and Gestalt Organization Using Spatial/Spatial Frequency Representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 1–12, Jan. 1990.

Richards81. J.A. Richards, D.A. Landgrebe, and P.H. Swain, "Pixel Labelling by Supervised Probabilistic Relaxation," *IEEE Transactions on Pattern Recognition and Machine Analysis*, vol. 3, no. 2, pp. 188–191, Mar. 1981.

Richards81a. J.A. Richards, D.A. Landgrebe, and P.H. Swain, "On the Accuracy of Pixel Relaxation Labelling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 4, pp. 303–309, Apr. 1981.

Richards86. J.A. Richards, *Remote Sensing Digital Image Analysis*. Berlin: Springer-Verlag, 1986.

Rosenfeld82. A. Rosenfeld, C.Y. Wan, and A.Y. Wu, "Multispectral Texture," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 12, no. 1, pp. 79–84, Jan. 1982.

Rosenfeld90. A. Rosenfeld, "Image Analysis and Computer Vision: 1989", *Computer Graphics and Image processing*, May 1990. (obtained electronically)[1]

Rumelhart86. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, D.E. Rumelhart and J.J. McClelland, Eds. MIT press, 1986, Ch. 8, pp. 318–360.

Rumelhart86a. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Back Propagation Errors," *Nature*, vol. 323, pp. 533–536, 9th Oct. 1986.

Schachter78. B.J. Schachter, A. Rosenfeld, and L.S. Davis, "Random Mosaic Models for Textures," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 8, no. 9, pp. 694–702, Sep. 1978.

Schowengerdt83. R.A. Schowengerdt, *Techniques for Image Processing and Classification in Remote Sensing*. Florida: Academic Press, 1983.

Settle91. J.J. Settle and N.A. Drake, "Linear Mixing and the Estimation of Ground Cover Proportions," *International Journal of Remote Sensing*, (preprint), 1991.

Sheffield85. C. Sheffield, "Selecting Band Combinations from Multispectral Data," *Photogrammetric Engineering and Remote Sensing*, vol. 51, no. 6, pp. 681–687, June 1985.

Sheldon90. R.A. Sheldon, "Satellite Image Analysis using Neural Networks," *Telematics and Informatics*, vol. 7, no. 3/4, pp. 431–439, 1990.

Shih83. E.H.H. Shih and R.A.S. Schowengerdt, "Classification of Arid Geomorphic Surfaces Using Landsat Spectral and Textural Features," *Photogrammetric Engineering and Remote Sensing*, vol. 49, no. 3, pp. 337–347, Mar. 1983.

Sietsma88. J. Sietsma and R.J.F. Dow, "Neural Net Pruning – Why and How.," in *Proc. IEEE Conference on Neural Networks, 1988*, 1988, pp. 211–218.

Sietsma91. J. Sietsma and R.J.F. Dow, "Creating Artificial Neural Networks That Generalize," *Neural Networks*, vol. 4, pp. 67–79, Jan. 1991.

Siew88. L.H. Siew, R. Hodgson, and E.J. Wood, "Texture Measures for Carpet Wear Assessment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 1, pp. 92–105, Jan. 1988.

Skrzypek87. J. Skrzypek and E. Mesrobian, "Textural Segmentation: Gestalt Heuristics as a Connectionist Hierarchy of Feature Detectors," in *Proc. Ninth Annual Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, 1987, pp. 1704–1706.

Smith90. M.O. Smith, J.B.A. Adams, and A.R. Gillespie, "Reference Endmembers for Spectral Mixture Analysis," in *Proc. 5th Australasian Remote Sensing Conference*, Perth, Western Australia, Oct. 1990, pp. 331–340.

Sontag90. E.D. Sontag, "On the Recognition Capabilities of Feedforward Nets," Tech. Rept. No. SYCON 90-03, Rutgers Centre for Systems and Control, Dept. Maths., Rutgers University, New Brunswick, NJ 08903, Apr. 1990.

---

[1] Bibliographys from 1984–1991 are publicly available in machine readable form from internet host: cs.dal.ca (129.173.4.5) at the Univerity of Maryland.

Sontag90a. E.D. Sontag, "Feedback Stabilization using Two-Hidden-Layer Nets," Tech. Rept. No. SYCON-90-11, Rutgers Centre for Systems and Control, Dept. Math., Rutgers University, New Brunswick, NL 08903, Oct. 1990.

Sontag91. E.D. Sontag, "Feedforward Nets for Interpolation and Classification," *Journal of Computer Systems Science*, vol. 45, 1992, preprint from Neuroprose archive, dated 1991.

Sun91. R. Sun, "Integrating Rules and Connectionism for Robust Reasoning," Tech. Rept. No. TR-CS-90-154, Com. Sci. Dept., Brandeis University, Walrham, MA 02254, Jan 1991.

Swain78. P.H. Swain, "Fundamentals of Pattern Recognition in Remote Sensing," in *Remote Sensing: The Quantitative Approach*, P.H. Swain and S.M. Davis, Eds. McGraw Hill, 1978, Ch. 3, pp. 137–187.

Swain80. P.H. Swain, H.J. Siegel, and B.W. Smith, "Contextual Classification of Multispectral Remote Sensing Data Using a Multiprocessor System," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 18, no. 2, pp. 197–203, Apr. 1980.

Swain81. P.H. Swain, S.B. Vaderman, and J.C. Tilton, "Context Classification of Multispectral Image Data," *Pattern Recognition*, vol. 13, no. 6, pp. 429–441, 1981.

Swain90. M.J. Swain, "Color Indexing," Tech. Rept. No. 360, University of Rochester, Computer Science, Rochester, New York, Nov. 1990.

Tilton81. J.C. Tilton and P.H. Swain, "Incorporating Spatial Context Into Statistical Classification of Multidimensional Image Data," Tech. Rept. No. SR-P1-04148, Purdue University, Laboratory for Applications of Remote Sensing, Indiana, Aug. 1981.

Tollenaere90. T. Tollenaere, "SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties," *Neural Networks*, vol. 3, pp. 561–573, 1990.

Tominaga88. S. Tominaga, "A Color Classification Algorithm for Color Images," in *Pattern Recognition, (Proc. 4th International Conference on Pattern Recognition, Cambridge, UK)*, J. Kittler, Ed, Mar. 1988, pp. 163–172.

Tribus88. M. Tribus, "An Engineer Looks at Bayes," in *Maximum-Entropy and Bayesian Methods in Science and Engineering*, G.J. Erickson and C.R. Smith, Eds. Kluwer Academic Publishers, 1988, pp. 31–52.

Unser86. M. Unser, "Sum and Difference Histograms for Texture Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 119–125, Jan. 1986.

Unser86b. M. Unser, "Local Linear Transforms for Texture Measurements," *Signal Processing*, vol. 11, pp. 61–79, 1986.

Unser89. M. Unser and M. Eden, "Multiresolution Feature Extraction and Selection for Texture Segmentation," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 11, no. 7, pp. 717–728, July 1989.

vanGool85. L. van Gool, P. de Waele, and A. Oosterlinck, "Texture Analysis Anno, 1983," *Computer Vision and Image Processing*, vol. 29, pp. 336–357, 1985.

Verlysen91. M. Verlysen and P. Jespers, "Precision of Computations in Analogue Neural Networks," in *VLSI design of Neural Networks*, U. Ramacher and U. Rückert, Eds. MA: Kluwer Academic, 1991, pp. 65–82.

Vickers82. A.L. Vickers and J.W. Modestino, "A Maximum Likelihood Approach to Texture Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, no. 1, pp. 61–68, Jan. 1982.

Vilnrotter86. F.M. Vilnrotter, R. Nevatia, and K.E. Price, "Structural Analysis of Natural Textures," *IEEE Transactions on Pattern Analysis Machine and Intelligence*, vol. 8, no. 1, pp. 721–741, Jan. 1986.

Visa90. A. Visa, "A Texture Classifier Based on Neural Network Principles," in *Proc. International Joint Conference on Neural Networks*, San Diego, 1990, pp. 491–496.

Voorhees87. H. Voorhees and T. Poggio, "Detecting Textons and Texture Boundaries in Natural Images," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1987, pp. 250–258.

Wang81. S. Wang, F.R.D. Velasco, A. Wu, and A. Rosenfeld, "Relative Effectiveness of Selected Texture Primitive Statistics for Texture Discrimination," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, pp. 360–370, May 1981.

Watson90. C.R. Watson and M. Cavaiuolo, "A Neural Accelerator," in *Ninth Australian Microelectronics Conference*, July 1990, (preprint).

Weszka76. J.S. Weszka, C.R. Dyer, and A. Rosenfeld, "A Comparative Study of Texture Measures of Terrain Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, no. 4, pp. 269–285, Apr. 1976.

Wharton87. S.W. Wharton, "A Spectral-Knowledge-Based Approach for Urban Land-Cover Discrimination," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 25, no. 3, pp. 272–282, May 1987.

Whatmough88. R.J. Whatmough, "Comment on Military GIS Article," *Australian Computer Journal*, vol. 20, no. 1, pp. 48, Feb. 1988.

Whatmough90. R.J. Whatmough, *Bias in the Calculation of PROC Distance*, (Personal Communication), Feb. 1990 .

White90. H. White, "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings," *Neural Networks*, vol. 3, pp. 535–549, 1990.

Widrow60. G. Widrow and M.E. Hoff, "Adaptive Switching Circuits", in *Convention Record Part 4, IRE Western Electronic Show and Convention*, 1960, pp. 96–104.

Widrow88. B. Widrow, R.G. Winter, and R.A. Baxter, "Layered Neural Nets for Pattern Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 7, pp. 1109–1118, July 1988.

Witkin84. A.P. Witkin, "Scale space filtering: a new approach to multi-scale description," in *Image Understanding 1984, Chapter 3*, S. Ullman and W. Richards, Eds. NJ: Ablek, 1984, pp. 79–95.

Woodcock87. C.E. Woodcock and A.H. Stahler, "The Factor of Scale in Remote Sensing," *Remote Sensing of the Environment*, vol. 21, pp. 311–332, 1987.

Wright89. W.A. Wright, "A Markov random field approach to data fusion and colour segmentation," *Image and Vision Computing*, vol. 7, no. 2, pp. 144–150, 1989.

Xie91. Y. Xie and M. Jabri, "Analysis of the effects of quantization in multilayer neural networks using statistical models," *Electronics Letters*, vol. 27, no. 13, pp. 1196–1198, June 1991.

Young74. T.Y. Young and T.W. Calvert, *Classification, Estimation and Pattern Recognition.* American Elsevier, 1974.

Zhang90. M.C. Zhang, R.M. Haralick, and J.B. Campbell, "Multispectral Image Context Classification Using Stochastic Relaxation," *IEEE Transactions on System Man and Cybernetics*, vol. 20, no. 1, pp. 128–140, Jan. 1990.

Zucker78. S.W. Zucker and J. Mohammed, "Analysis of Probabilistic Relaxation Labelling Processes," *IEEE Conference on Pattern Recognition and Image Processing, Chicago, Il.*, pp. 307–312, May 1978.

Attachment A

Weights evolving during training.

The attached floppy disk contains a sequence of images that represent the weights in an MLP of structure 4-25-6-5 during the learning phase. Weights were sampled every 800 weight adjustments, and there are a total of 68 frames representing a total of 53,600 weight adjustments (the first frame is after no adjustments).

Hardware requirements: Apple Macintosh Computer, able to display 256 colours.

Software requirements: Operating system 6.0.5 or later.

Disk contains "Weights-Demo" and a copy of the NIH program Image1.43.

Viewing the demonstration: Insert the disk and double click on the document "Weights-Demo". You may wish to re-size the resulting display. Click in the display to stop the animation. Select "Options" menu and check "Scale to fit window". Then click and drag on the bottom right hand corner of the window until it fills the screen. The aspect ratio of the window will be maintained so it may not fill the whole screen.

The display uses colours to represent the weight values. Green corresponds to zero, blue-purple corresponds to -7.0 and red corresponds to +7.0. The continuum of colours is visible in the LUT window to the left.

The first seven rows of the image represent weights from the fixed bias plus 6 nodes of layer 2 arranged in columns to correspond to the 5 nodes of layer 3. The next 26 rows represent weights from the fixed bias plus 25 nodes in the first active layer arranged in 6 columns corresponding to the nodes in layer 2. The bottom 5 rows represent weights from the 4 multi-spectral inputs plus fixed bias arranged in 25 columns corresponding to the nodes of layer 1. The unused area of the images is set to zero and appears green.

To begin animation, select "Stacks" menu and choose option "Animate". A white square should move from top right to bottom right to indicate progress through the sequence of images.

It can be observed that weights near the output of the MLP "learn" quickly compared to weights near the input of the MLP. It is also evident that few nodes are ever trained to have opposite polarity to their starting values. It appears that nodes that produce information that is not useful, or wrong, are simply ignored. Those that provide useful or credible information provide strong inputs to the next layer. The parallels with human organizations are inescapable.

To finish, select "File" menu and choose option "Quit".